National Research Council Canada
Conseil national de recherches Canada

Division of Electrical Engineering
Division de génie électrique

ERB-985

Polyhedra Recognition by Hypothesis Accumulation from Range Data

M. Dhome and T. Kasvand

May 1986

Canada

NRCC No. 26605

86-06447/

Copyright 1986 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Copyright 1986 par
Conseil national de recherches du Canada

Il est permis de citer de courts extraits et de reproduire des figures ou tableaux du présent rapport, à condition d'en identifier clairement la source.

Additional copies are available free of charge from:

Editorial Office, Room 301
Division of Electrical Engineering
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R8

Des exemplaires supplémentaires gratuits peuvent être obtenus à l'adresse suivante:

Bureau des publications, Pièce 301
Division de génie électrique
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R8

POLYHEDRA RECOGNITION BY HYPOTHESIS ACCUMULATION FROM RANGE DATA

M. Dhome and T. Kasvand

ERB-985

NRCC No. 25655

MAY 1986

TABLE OF CONTENTS

Page

ABSTRACT/RÉSUMÉ v
I. INTRODUCTION 1
II. PRESENTATION OF THE METHOD 2
1. Modelling the Objects 5
2. Basic Elements from the Studied Image 6
3. Matching of Pairs of Adjacent Surfaces 6
4. Search for the View Axis 6
5. Search for the View Orientation 7
6. Search for the Model Center 8
7. Model Transform and Matching with Range Data 9
8. Selection of the Best Locations of the Studied Model 10
9. Analysis of Object Overlap 11
III. RESULTS 12
IV. CONCLUSION 25
ACKNOWLEDGEMENTS 26
REFERENCES 26
APPENDIX 29
A. "CREATIONOBJET" PROGRAM 30
B. "LECTUREOBJET" PROGRAM 34
C. "VISUALISATIONOBJET" PROGRAM 39
D. "MODELISATIONOBJET" PROGRAM 46
E. "CREATIONIMAGE" PROGRAM 50
F. "LECTURESCENE" PROGRAM 60
G. "ANALYSESCENE" PROGRAM 86
H. "IMPRIMEIMAGE" PROGRAM 119
I. "PROJECTIONOBJET" PROGRAM 133
J. "UTILITAIRES" FILE 140

FIGURES

1. Hierarchical scheme used to find the triplets 3
2. (a) The "CAD-Model" of an object. (b) A "Local Geometrical Pattern" used in the "LGP-Model" of the same object. 5
3. A "Local Geometrical Pattern" extracted from the scene 6
4. (a) and (b) Construction of two straight lines. (c) Possible position of the model center 8
5. Visualisation of the three rotations R_x, R_y, and R_z 9
6. (a) Visualisation of a scene object. (b) Overlapping of the object by a bigger model. (c) Overlapping of the object by a smaller model 11

7. Visualisation of the 11 objects used for the experiments 12
8. (a) An artificial scene. (b) Planar surfaces extracted from this artificial scene 14
9. (a) Range data. (b) Visualization of normals at each point of the scene. (c) Non-vertical planar surfaces extracted from the range data. (d) Overlapping of normal images and selected models 17
10. (a) Range data. (b) Visualization of normals at each point of the scene. (c) Non-vertical planar surfaces extracted from the range data. (d) Overlapping of normals image and selected models 23
A-1. CAD-Model of the object No. 2 35
A-2. Visualisation of the objects No. 4, 10, 11 presented in Fig. 8a 40
A-3. Output files of the "CREATIONIMAGE" program. (a) Range data of an artificial image. (b) Location of the different polyhedra. (c) Location of the different plane surfaces. (d) Adjacency of the different plane surfaces 51
A-4. Input files of the "LECTURESCENE" program. (a) Range data. b) Location of the different groups of overlapping objects. (c) Location of the different plane surfaces. (d) Adjacency of the different plane surfaces 62
A-5. Output files of the "LECTURESCENE" program. (a) New labelling of the plane surfaces. (b) Location of the vertical surfaces. (c) Modified range data 66
A-6. Input files of the "ANALYSESCENE" program. (a) Location of the different groups of overlapping objects. (b) New labelling of the plane surfaces. (c) Modified range data. 88
A-7. Result of the analysis of the scene corresponding to Fig. A-6 . 91
A-8. Output files of the "IMPRIMEIMAGE" program. (a) Range data. (b) Normal at each pixels of the scene. c) Value at each pixel of the scene. (d) Superposition of two images (Figs. A-8c and A-9) 121
A-9. Visualisation of the result of the analysis of the scene corresponding to the Fig. A-6 134

TABLES

1. Results of the analysis of Fig. 8 13
2. (a) Results of the analysis of the right group of overlapping objects of the scene presented in Fig. 9. (b) Results of the analysis of the left group of overlapping objects of the scene presented in Fig. 9 16
3. Results of the analysis of the scene presented in Fig. 10. . . . 25

ABSTRACT

A new parallel method for the recognition of polyhedra in range data, based on a hypothesis accumulation scheme, is presented. The different objects to be recognized are modeled by a set of local geometrical patterns. Elements of the same nature are extracted from the scene. For the recognition of an object similar to a model, local scene and model patterns having the same geometrical characteristics are matched. For each of these possible matchings, the geometric transformation (i.e., rotations + translations) which allows the exact overlap of the model element with the one from the scene is computed. This transformation permits the establishment of a hypothesis on the location of the object in the scene and the determination of a point in the transformation space. The presence of an object similar to the model involves several compatible hypothesis and creates a compact cluster in the transformation space. The recognition of this object is based on the detection of this cluster. The cluster coordinates indicate the value of the rotations and the translations to be applied to the model such that it corresponds to the object in the scene. The exact location of this object is given by the transformed model.

RÉSUMÉ

Une nouvelle méthode pour la reconnaissance de polyèdres à partir de données télémétriques est présentée. Elle est basée sur un processus d'accumulation d'hypothèses. Les différents objets à reconnaître sont modelisés par un ensemble de formes géométriques locales. Des éléments de même nature sont extraits de la scène étudiée. Puis, pour aboutir à la reconnaissance d'objets identiques à un modèle donné, les formes locales de ce modèle sont mises en correspondance avec celles de la scène ayant des caractéristiques géométriques identiques. Pour chacune de ces mises en correspondance, la transformation géométrique (rotations + translations) qui permet le recouvrement exact de l'élément de la scène par celui du modèle est calculée. Cette transformation permet d'établir une hypothèse sur la localisation d'un objet dans la scène et de déterminer un point unique dans l'espace des transformations. La présence, dans la scène, d'un objet identique au modèle donnera naissance à plusieurs hypothèses compatibles et entrainera la determination d'un ensemble compact de points dans l'espace des transformations. La reconnaissance de cet objet sera fondée sur la détection de cet ensemble de points. La position de ce dernier permettra de définir les rotations et les translations à appliquer au modèle pour qu'il se superpose à l'objet dans la scène. La position exact de cet objet sera donc identique à celle du modèle après transformation.

POLYHEDRA RECOGNITION BY HYPOTHESIS ACCUMULATION FROM RANGE DATA

M. Dhome and T. Kasvand

I. INTRODUCTION

The recognition of 3-D objects and the determination of their position in space are two main problems for robot vision systems. In the past, only gray level pictures were available as input data. Many methods for object recognition using these images have been presented in the literature (1-7). All require edge detection of the objects in the scene as the recognition is based on matching these edges with those of the 3-D models. Unfortunately, for many reasons (shadows, brightness of object surfaces, modification of lighting conditions, ...), edge detection is not always reliable.

To overcome this problem, range finders, which measure directly the three-dimensional coordinates, have recently been studied (8). To readers familiar with traditional image processing, the range at each point of the image can be thought of as "gray level." From these new data, it is possible to extract characteristics of the different surfaces for each object of the scene (9-13). Using this knowledge, new methods for the recognition of 3-D objects have been developed (14-22).

In industrial applications, object recognition must be as fast as possible. Algorithms compatible with parallel computer structures are therefore advantageous. However, most methods proposed previously are not particularly well suited for parallel computation. The goal of this paper is to present a parallel algorithm for the recognition of polyhedra. It is based on the principle of the Hough transform (16) and more particularly on the generalization given by Ballard (2). Although the Hough transform is used widely and successfully in the case of bidimensional images (2, 4, 6, 7, 23), few people have adapted it to tridimensional data. The exceptions have included Henderson and Fai (24), who worked on the recognition of very simple objects such as cubes, and Tanaka et al. (20), who proposed a method for more general polyhedra. Unfortunately, Tanaka et al.'s results concerned artificial images containing an isolated object.

The algorithm described below is not affected by occlusion and can correctly analyse real scenes with overlapping polyhedra.

II. PRESENTATION OF THE METHOD

Generally, the methods based on the principle of the generalized Hough transform (2, 4, 6, 7) use an accumulator array. Each dimension of this array corresponds to one possible geometrical transformation (i.e., rotation, translation, scale variation) that an object can undergo. For each matching between the scene and model elements, the global transformation which permits the superposition of the model element with one from the scene is computed and the corresponding point of the accumulator array is incremented. At the end of the accumulation procedure, the points which have the maximum values correspond to the highly probable occurrences of scene objects similar to the model.

In 3-D space, an object has six degrees of freedom which correspond to one rotation and one translation around and along each axis of the reference system. In these conditions, although we consider that the scale of the analyzed scene is known a priori, the recognition of objects by the generalized Hough transform requires a six-dimensional array. The size of this array required, for the determination with a good accuracy of object position (e.g., one degree rotation and one pixel translation), is too large to be practical. For this reason, an accumulator array is not used; however, the method does preserve the principle of hypothesis accumulation.

In this approach, the different objects to be recognized are modeled by a set of local geometrical patterns. For polyhedron recognition, the chosen local patterns are the pairs of adjacent surfaces. From the analyzed scene, the plane surfaces are extracted (for more details see companion paper (13) and a set of elements of the same type are built. Local scene and model patterns having the same geometrical characteristics (i.e., equality of the angle between the normals of the adjacent surfaces) are matched in the search for objects similar to a model. Each of these matchings permits the generation of a hypothesis on the location of an object in the scene. This location corresponds to the model position in space which allows the exact overlap of the pair of model surfaces with the pair of scene surfaces. Three parameters characterizing this model position are stored. These parameters include the model view axis, the model orientation around the view axis, and the model center position. The presence in the scene of an object similar to the model involves the generation of several compatible hypotheses. The recognition is simply based on the detection of the large sets of agreeing hypotheses. These are found by a clustering program working on the stored parameters. To minimize the amount of computation and to simplify the search for the corresponding clusters in the three different parameter spaces, the method uses the following hierarchical scheme (see Fig. 1):

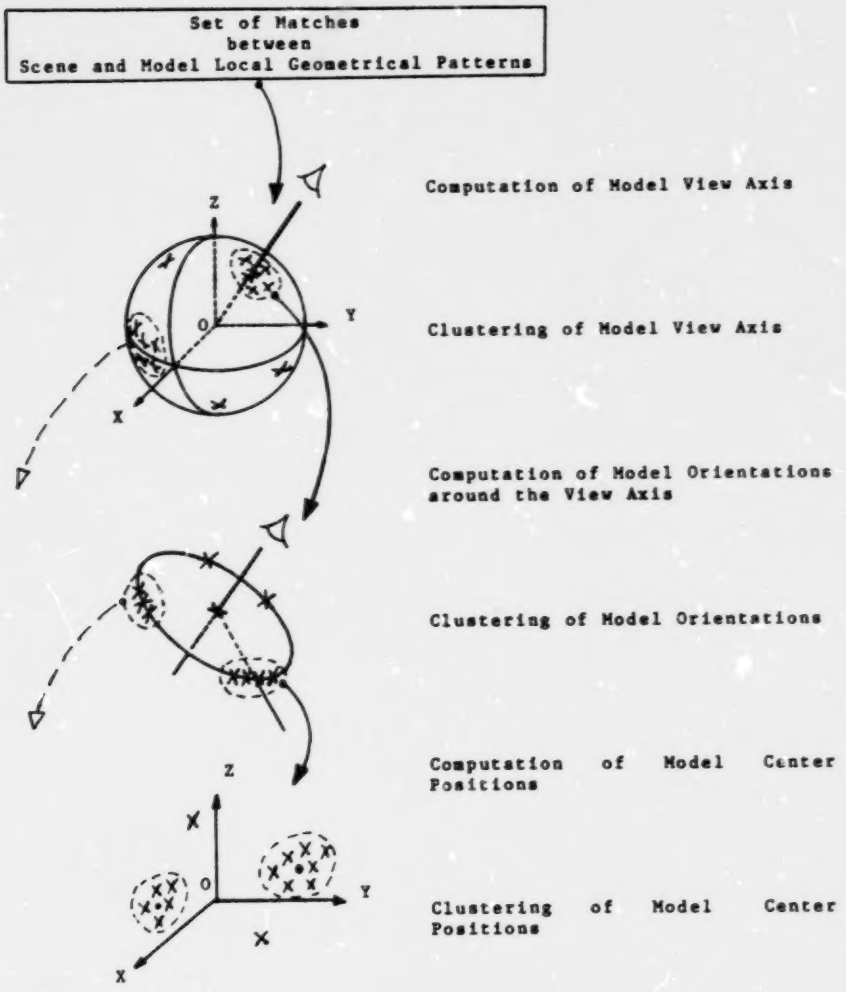


Figure 1. Hierarchical scheme used to find the triplets (view axis, orientation around the view axis, position of the center of the model).

- first, from all the possible matches between the pairs of adjacent surfaces of the model and those of the studied scene, the algorithm looks for the view axis of the model.

- then, for each view axis found, compatible matches are selected and, from them, possible orientations of the model around this axis are calculated.

- finally, for each pair of corresponding view axes and orientations, the algorithm computes, from the compatible matches, the possible positions of the reference center of the model.

The knowledge of one triplet (view axis, orientation around the view axis, position of the center) allows the determination of the value of the rotations and the translations to be applied to the model such that it corresponds to the object in the scene.

Similarities between local patterns of a model(s) can involve the detection of different positions for this model(s) corresponding to the same object of the scene. The selection of the good solution(s) (i.e., model + location) is a function of the number of elements in the clusters (i.e., number of agreeing hypotheses) and the value of a quality index which measures the agreement between the scene (i.e., range data) and the transformed models (i.e., rotated and translated).

Generally, the analyzed scene contains many objects where some may overlap. To reduce the number of matches of local pattern, the method considers each separate group of object(s) (these groups are differentiated by thresholding the range data during pre-processing (for more details see ref. 13) and successively applies the different models on each of them.

The method corresponds to the algorithm written below (the different sub-programs are described in more detail in the following paragraphs).

```
BEGIN
FOR Object Group Index:= 1 TO Object Group Number DO
  BEGIN
  FOR Model Index:= 1 TO Model Number DO
    BEGIN
    Matching of Pairs of Adjacent Surfaces;
    Search for the View Axis;
    FOR View Axis Index:= 1 TO View Axis Number DO
      BEGIN
      Search for the View Orientation;
      FOR Orientation Index:= 1 TO Orientation Number DO
        BEGIN
        Search for the Model Center;
        FOR Model Center Index:= 1 TO Model Center Number DO
          Model Transform and Matching with Range Data;
        END;
      END;
    Selection of the Best Locations of the Studied Model;
    END;
  Analysis of Objects Overlap;
  END;
END.
```

Note: It is important to see that at each step of the five loops the algorithm works in an independent way, making it compatible with parallel computer structures at several levels.

1. Modelling the Objects

For each object, two different models are built:

- the first one is used only for the geometrical representation of the object (see Fig. 2a). It can be generated by a "Computer Aided Design system" and therefore is called the "CAD-Model." It contains a list of the vertices of the polyhedron and, for each surface, the set of vertex list indices which belong to it.

- the second is used for matching with the analyzed scene. It is represented by a list of Pascal records corresponding to the "Local Geometrical Patterns" and called "LGP-Model." Each record references a pair of adjacent surfaces and contains five parameters which describe the location of these surfaces with reference to the center of the model. This center is at the origin of the reference system where the CAD-Model is defined and can be arbitrarily chosen.

These five parameters are (see Fig. 2b):

$A_{m_{ij}}$ - angle between the normals of the surfaces S_{m_i} and S_{m_j}

N_{m_i}, N_{m_j} - normals of the surfaces S_{m_i} and S_{m_j}

D_{m_i}, D_{m_j} - orthogonal distances between the model center and the surfaces S_{m_i} and S_{m_j}

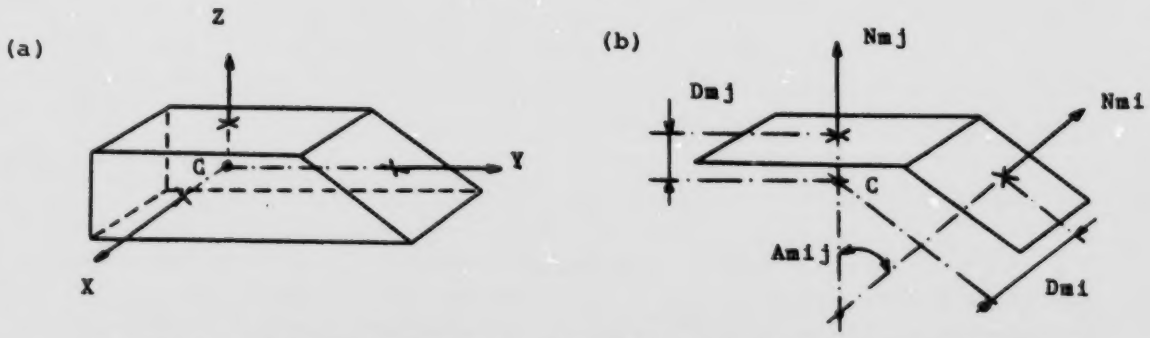


Figure 2. (a) The "CAD-Model" of an object. (b) A "Local Geometrical Pattern" used in the "LGP-Model" of the same object.

2. Basic Elements from the Studied Image

From the studied image, the planar surfaces are extracted (see companion paper (13)) and a second list of records is built. Each record corresponds to a pair of adjacent surfaces and contains five parameters which are (see Fig. 3):

$A_{i_{k\ell}}$ - angle between the normals of the surfaces S_{i_k} and S_{i_ℓ}

N_{i_k}, N_{i_ℓ} - normals of the surfaces S_{i_k} and S_{i_ℓ}

C_{i_k}, C_{i_ℓ} - gravity centers of the surfaces S_{i_k} and S_{i_ℓ}

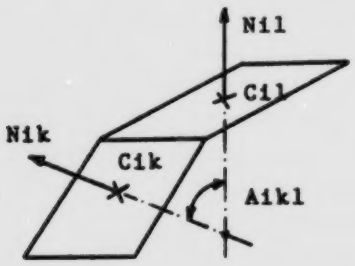


Figure 3. A "Local Geometrical Pattern" extracted from the scene.

3. Matching of Pairs of Adjacent Surfaces

Since the angle between the normals of two adjacent surfaces is invariant to rotation and translation, it is used as the geometric characteristic to control the match between a model surface pair (S_{m_i}, S_{m_j}) and a scene surface pair (S_{i_k}, S_{i_ℓ}). So (S_{m_i}, S_{m_j}) and (S_{i_k}, S_{i_ℓ}) are matched if and only if $A_{m_{ij}}$ is equal to $A_{i_{k\ell}}$. In practice the equality is not strict and we accept a small angular variation ($A_{m_{ij}} = A_{i_{k\ell}} \pm \varepsilon$) but in this case the scene surface normals are symmetrically modified such that $A_{m_{ij}} = A_{i_{k\ell}}$.

4. Search for the View Axis

The coordinate system of the studied scene is a left-handed system where the Z-axis corresponds to the elevation (range finder axis). So, all the objects in the scene are viewed by the camera along the Z-axis. This sub-program searches for the axis of the model reference system equivalent to the Z-axis of the scene.

For each possible matching of two pairs of adjacent surfaces (S_{m_i}, S_{m_j}), (S_{i_k}, S_{i_ℓ}) the direction vector of the model view axis (V_a) is computed such that the projections of the model surface normals (N_{m_i}, N_{m_j}) on V_a are respectively equal to the projection of the scene surface normals (N_{i_k}, N_{i_ℓ}) on V_z (scene Z-axis vector). This computation is very simple since it corresponds to the following system of three equations with three unknowns:

$$N_{m_i} \cdot V_a = N_{i_k} \cdot V_z$$

$$N_{m_j} \cdot V_a = N_{i_\ell} \cdot V_z$$

$$N_{m_p} \cdot V_a = N_{i_q} \cdot V_z$$

where "$\cdot$" is the scalar product of two vectors and N_{m_p}, N_{i_q} are respectively equal to the cross product of the vectors N_{m_i}, N_{m_j} and N_{i_k}, N_{i_ℓ} $(N_{m_p} = N_{m_i} \times N_{m_j}, N_{i_q} = N_{i_k} \times N_{i_\ell})$.

When the vector V_a is calculated and stored for each match, the possible view axes are detected with a clustering program. This program corresponds to the "Furthest-Neighbor Algorithm" and, more precisely, to the "Complete-Linkage Algorithm" because it is halted when the distance between nearest clusters exceeds an arbitrary threshold (for more details see ref. 25, pages 228-233). The distance between two view axes is the angle included between the corresponding vectors. Only the clusters containing more than three vectors are selected and on each of them the average axis is computed.

5. Search for the View Orientation

By convention, all the objects of the analyzed scene are oriented along the X-axis. The goal of this sub-program is to find the axis of the model reference system equivalent to the X-axis of the scene. The direction vector (V_o) of this equivalent axis indicates the "orientation of view" around the view axis. This sub-program works for each selected view axis using only the matchings which have generated points in the corresponding cluster.

For a possible matching of two pairs of surfaces (S_{m_i}, S_{m_j}), (S_{i_k}, S_{i_ℓ}) the desired vector V_o is defined such that the projections of the model surface normals (N_{m_i}, N_{m_j}) on V_o are respectively equal to the projections of the scene surface normals (N_{i_k}, N_{i_ℓ}) on V_x (scene X-axis

vector). Thus, V_o is calculated in the same way as V_a in the previous paragraph and stored for all the compatible matchings with the studied view axis. The possible view orientations are found by using the same clustering algorithm.

6. Search for the Model Center

The search for the model center position is done twice:

- first, for each matching of adjacent surfaces pairs (S_{m_i}, S_{m_j}), (S_{i_k}, S_{i_ℓ}) according to the studied view axis and orientation, the equation of a particular straight line passing through the center of the model is calculated. This straight line is defined by the intersection of two planes which are parallel to the surfaces S_{i_k} and S_{i_ℓ} and situated at distances D_{m_i} and D_{m_j} from the points C_{i_k} and C_{i_ℓ} (see Fig. 4 a and b).

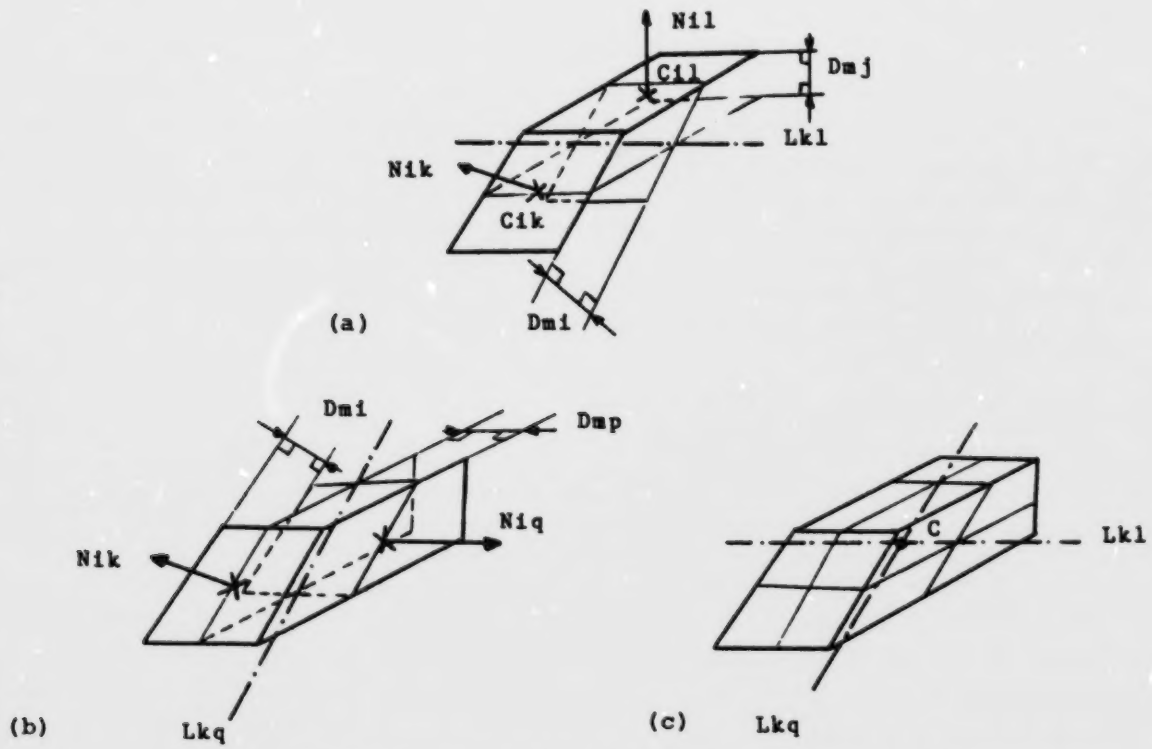


Figure 4. (a) and (b) Construction of two straight lines. (c) Possible position of the model center.

- then, among these straight lines, each combination of two non-parallel lines permits the generation of a hypothesis on the position of the model center in the analyzed scene (see Fig. 4c). If the lines are secant, this position corresponds to the intersection point; otherwise it corresponds to the nearest equidistant point of these two lines.

To find the highly probable occurrences of the model center, the set of the found points is also processed by the same clustering algorithm, but using the Euclidian distance between two points.

7. Model Transform and Matching with Range Data

Knowing the view axis, the view orientation and the model center position, it is possible to define the six transformations, in the model reference system, which will make the model correspond with the object in the scene. It is only necessary to compute (see Fig. 5):

- the rotation (R_x) around the X-axis which brings the view axis into the plane XOZ,

- the rotation (R_y) around the Y-axis which aligns the transformed view axis (rotated by R_x) with the Z-axis,

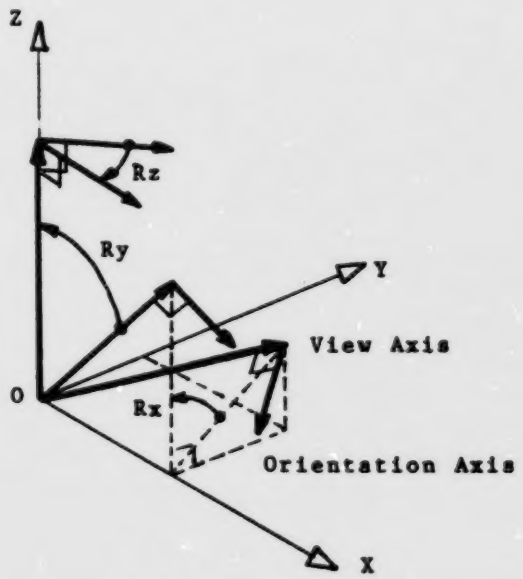


Figure 5. Visualisation of the three rotations R_x, R_y, and R_z.

- the rotation (R_z) around the Z-axis which aligns the transformed view orientation axis (rotated by R_x and R_y) into a position parallel to the X-axis,

- the three translations (T_x, T_y, T_z) corresponding to the coordinates of the calculated model center because the CAD-Model reference system is centered at this point.

These six transformations correspond to a possible location in the scene for an object identical tc the studied model. But some similarities between the local geometrical patterns of the model(s) can involve the detection of several locations for this model(s) corresponding to the same object in the scene. Thus, it is necessary to have some criteria to select the good solution(s). Among them we use a quality index which measures the agreement between the scene and the location of the model. This means that the six transformations must be applied to the CAD-Model and for each point of the projection of the transformed CAD-Model onto the XY-plane the elevation must be calculated. Then the quality index is computed as the sum of the two following average errors:

- the average error between the range data and the transformed CAD-Model for all the points of the XY-plane model projection (this average penalizes models bigger than the real object in the scene, see Fig. 6b)

- the average error between the range data and the transformed CAD-Model for all points in the XY-plane belonging to one surface of the scene matched with one of the model (this average penalizes the model smaller than the real object in the scene, see Fig. 6c)

8. Selection of the Best Locations of the Studied Model

Among all the locations found for the studied model, only the best one(s) must be retained. This selection is made by using sequentially the following three criteria:

- the set of scene surfaces matched with the model surfaces. (This criterion permits creation of a partition of locations. The locations which correspond to sets of scene surfaces having non-empty intersection are grouped within the same sub-set. So, after this classification, each sub-set is related to a different object of the scene.)

- the number of compatible hypotheses for a location which is equal to the number of elements in the final cluster during the search for

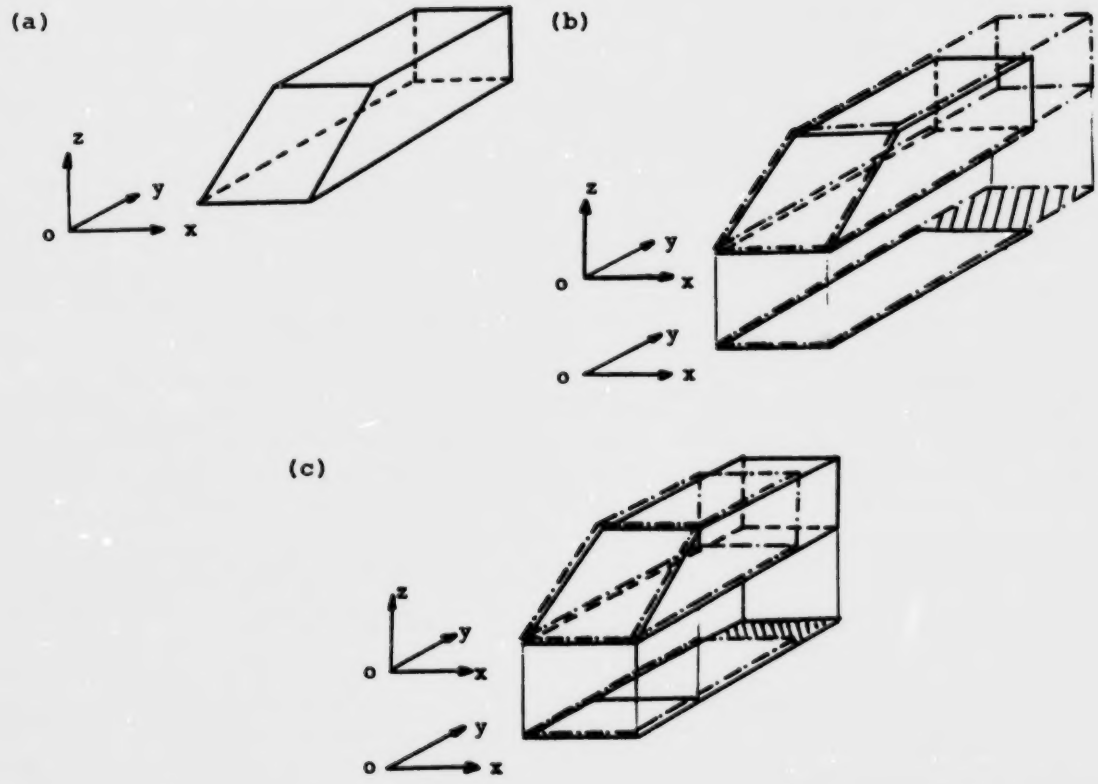


Figure 6. (a) Visualisation of a scene object. (b) Overlapping of the object (-) by a bigger model (-•-). The agreement is not good in the dashed area. (c) Overlapping of the object (-) by a smaller model (-••-). The agreement is not good in the dashed area.

the model center. (In each sub-set, the location having more compatible hypotheses is selected.)

- the quality index of the agreement of the location of the model with the initial scene. (After using the previous criterion, if several locations are retained in the same sub-set because they have the same number of compatible hypotheses, the one having the best (smallest) quality index is finally selected).

9. Analysis of Object Overlap

At this level of the algorithm, several locations of differents models are retained for each group of overlapping objects in the scene. The goal of this sub-program is to select the good solution(s) (i.e., model + location) and to understand the overlap of the different objects. For this, the following procedure is used:

- first, the solution having the best quality index is selected to correspond to the top object in the overlap.

- then, the quality index of all the other solutions are re-evaluated as described in the seventh paragraph of the second section, but without taking account of the points belonging to the projection onto the XY-plane of the selected model. The solutions having all the points of their own XY-plane projection included in the XY-plane projection of the selected model are cancelled. In this way, of the two locations of the different models corresponding to the same scene object, only the one having the best quality index is retained.

This procedure is reactivated until a possible solution exists for the subsequent objects from the top to the bottom of the group but at each step the quality index is recalculated without taking into account the points belonging to the XY-plane projection of all the previously selected models.

III. RESULTS

The proposed method has been tested on artificial and real scenes containing some of the 11 objects represented by their CAD-Model in Fig. 7. Note the important similarities among some of them. In all the cases

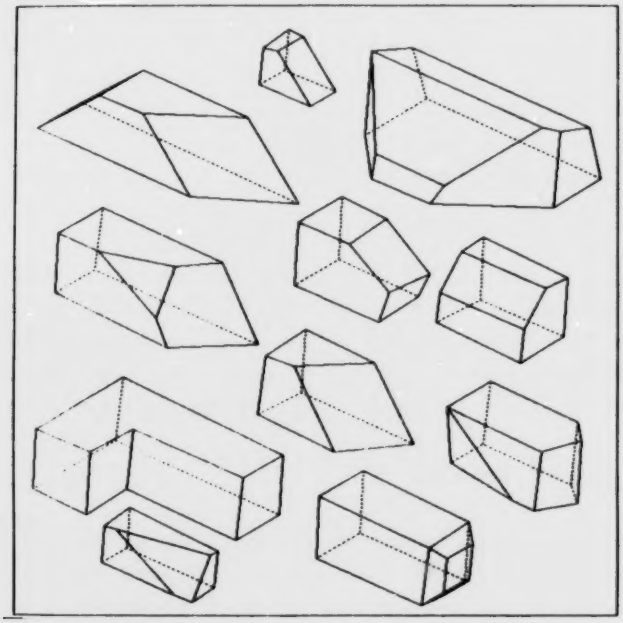


Figure 7. Visualisation of the 11 objects used for the experiments.

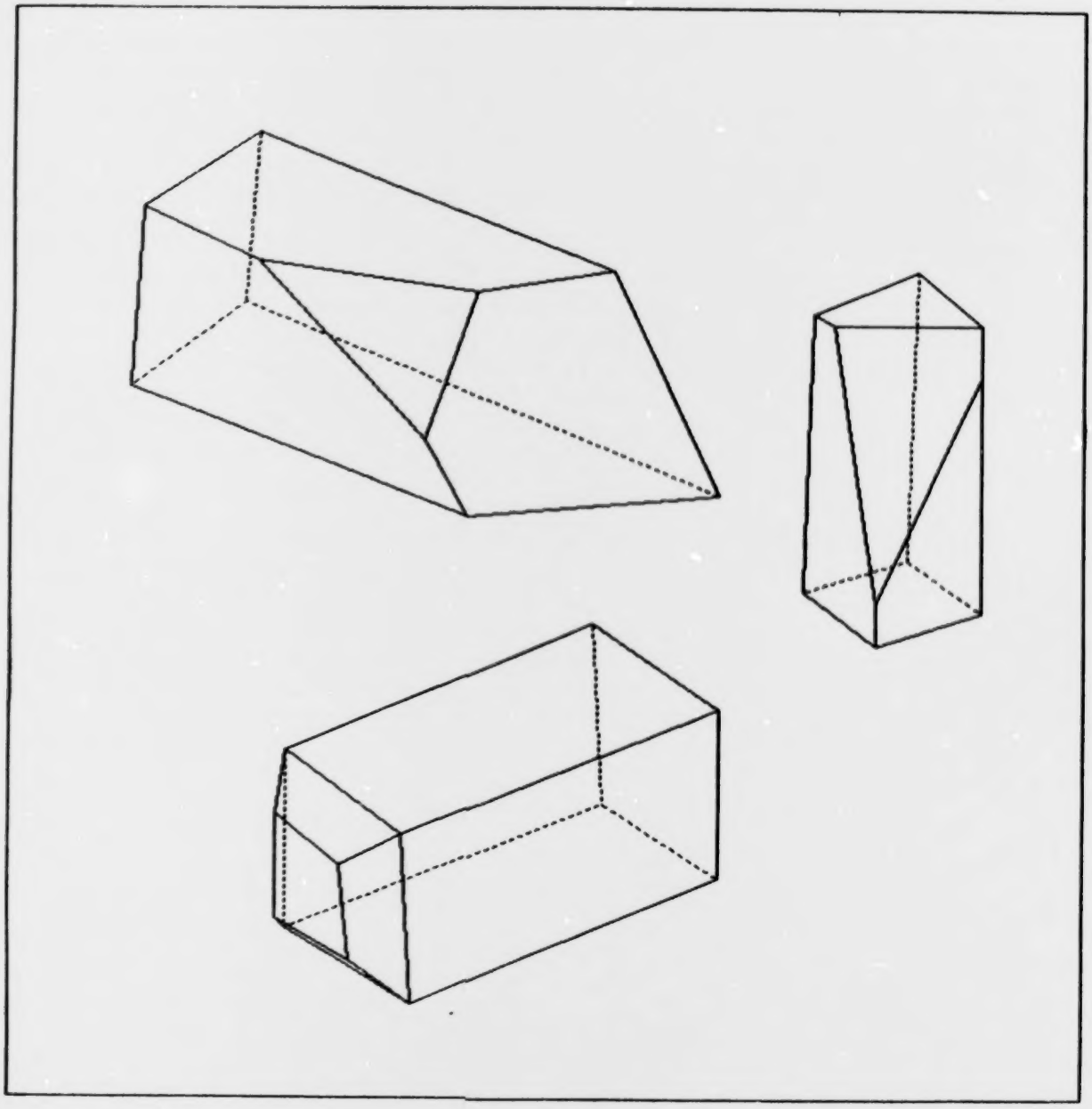
presented, the analysis of the scene has been made by using the 11 distinct models. The size of the used images is 128 pixels × 128 pixels and the vertical and horizontal distance between two pixels corresponds approximatively to 1 mm in the real scene. Moreover, the allowed angular variation for the matching of pairs of adjacent surfaces was equal to 10° and the thresholds used by the clustering program to find the view axis, the view orientations, and the model centers was respectively equal to 15°, 12°, and 6 mm.

The first result was obtained by the analysis of an artificial scene (Fig. 8) which contains three polyhedra. Known rotations and translations around and along the XYZ-axis were applied to these objects. The goal of this experiment was to recognize these three objects and evaluate the values of their geometrical transformations. The results are presented in Table 1. In this case, as in other similar tests, the objects are recognized and both rotations and translations are determinated with very good accuracy.

For the second and the third examples, the method has been applied to real scenes. The range data are produced by the range finder of the Electronic Engineering Section of the National Research Council of Canada, and are presented in Figs. 9a and 10a. The characteristics of this camera, described in more detail in ref. 8, result in some points of the scene (dependent on the position of the contained objects) not being visible from the camera. However, to fill up these "blind area," the range finder generates a false planar surfaces. However, these surfaces do not represent a

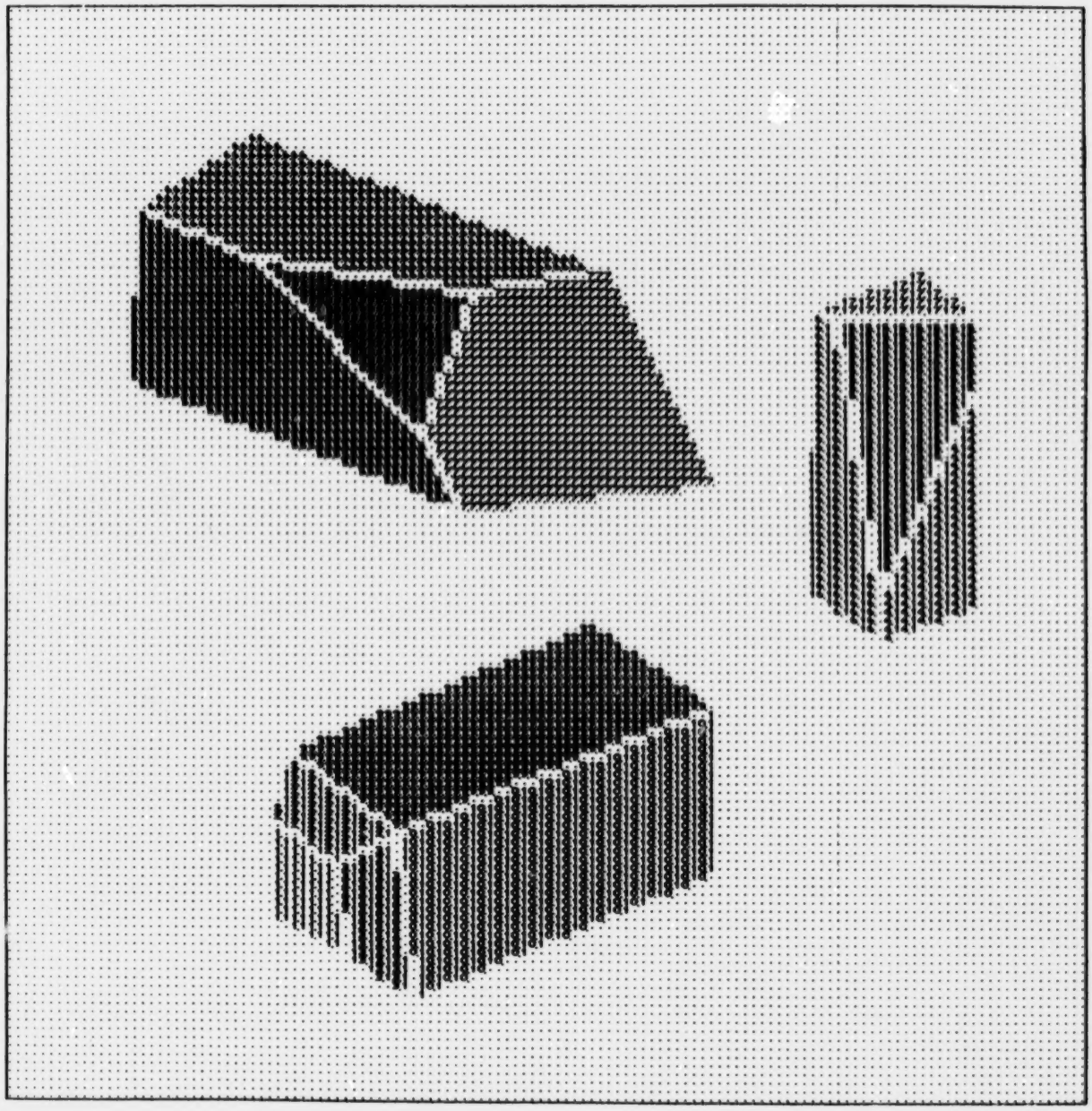
Table 1. Results of the analysis of Fig. 8.

Scene object	Found model	Rotation (°)			Translation (mm)		
			Applied	Found		Applied	Found
4	4	R_x	45	45	T_x	30	30
		R_y	-45	-45	T_y	35	35
		R_z	-57	-57	T_z	40	40
10	10	R_x	45	45	T_x	50	50
		R_y	-45	-45	T_y	85	85
		R_z	54	54	T_z	45	45
11	11	R_x	45	45	T_x	75	75
		R_y	45	45	T_y	45	45
		R_z	-123	-123	T_z	50	50



(a)

Figure 8. (a) An artificial scene. (b) Planar surfaces extracted from this artificial scene.



(b)

Figure 8 (Concluded).

handicap for the method. The normals at each pixel of these images, shown in Figs. 9b and 10b, give to readers an easier visual interpretation of the scenes. The planar surfaces extracted by the pre-processing step correspond to Figs. 9c and 10c, but only the non-vertical ones are presented. To appreciate the results of the method, the selected models are rotated, translated, and projected on the images of the normals (Figs. 9d and 10d) and the resulting values of the rotations, translations, and quality index are noted in Tables 2a, 2b, and 3.

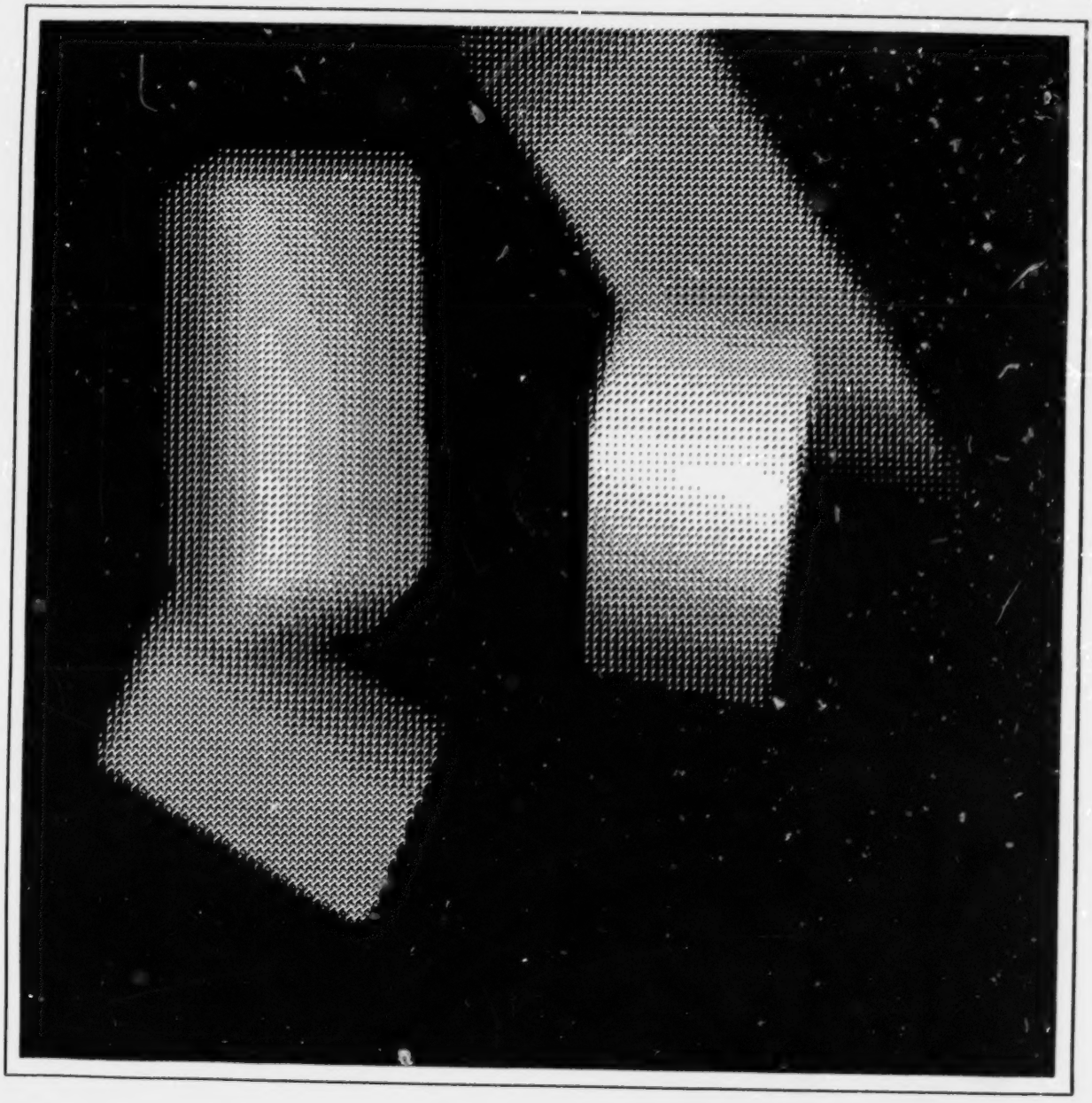
These results shows that the correct models have been selected for each object of the scene and that the calculated locations for these models correspond to the real positions of the objects as shown by the small value of the quality index.

Table 2a. Results of the analysis of the right group of overlapping objects of the scene presented in Fig. 9.

	Object	Model	Rotation (°)		Translation (mm)		Index (mm)
Top	5	5	R_x	-129	T_x	42	0.41
			R_y	-2	T_y	71	
			R_z	79	T_z	15	
	1	1	R_x	0	T_x	19	0.82
			R_y	0	T_y	71	
Bottom			R_z	-52	T_z	9	

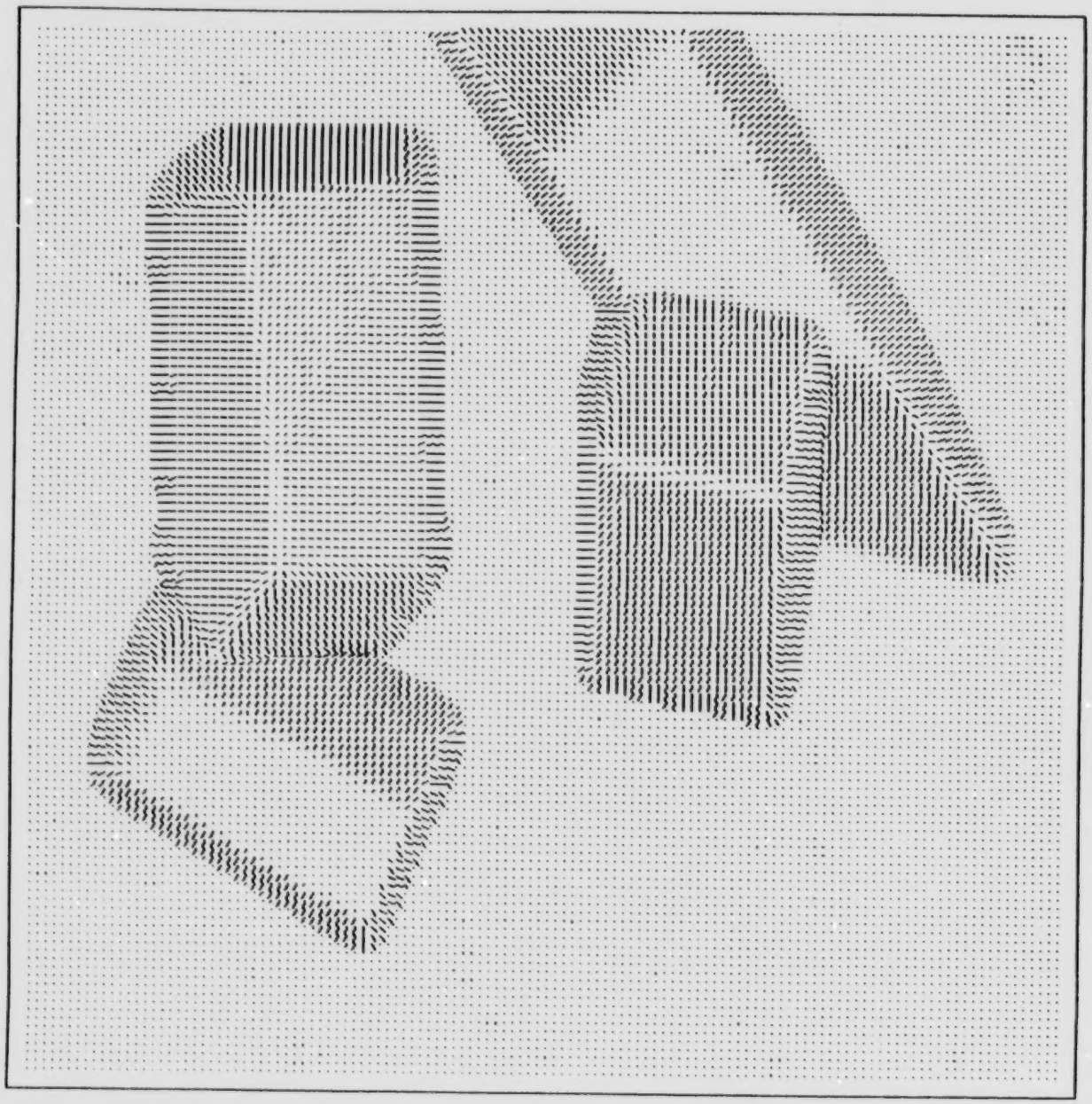
Table 2b. Results of the analysis of the left group of overlapping objects of the scene presented in Fig. 9.

	Object	Model	Rotation (°)		Translation (mm)		Index(mm)
Top	9	9	R_x	174	T_x	26	0.32
			R_y	-57	T_y	22	
			R_z	97	T_z	13	
	6	6	R_x	-176	T_x	66	0.99
			R_y	-89	T_y	15	
Bottom			R_z	151	T_z	3	



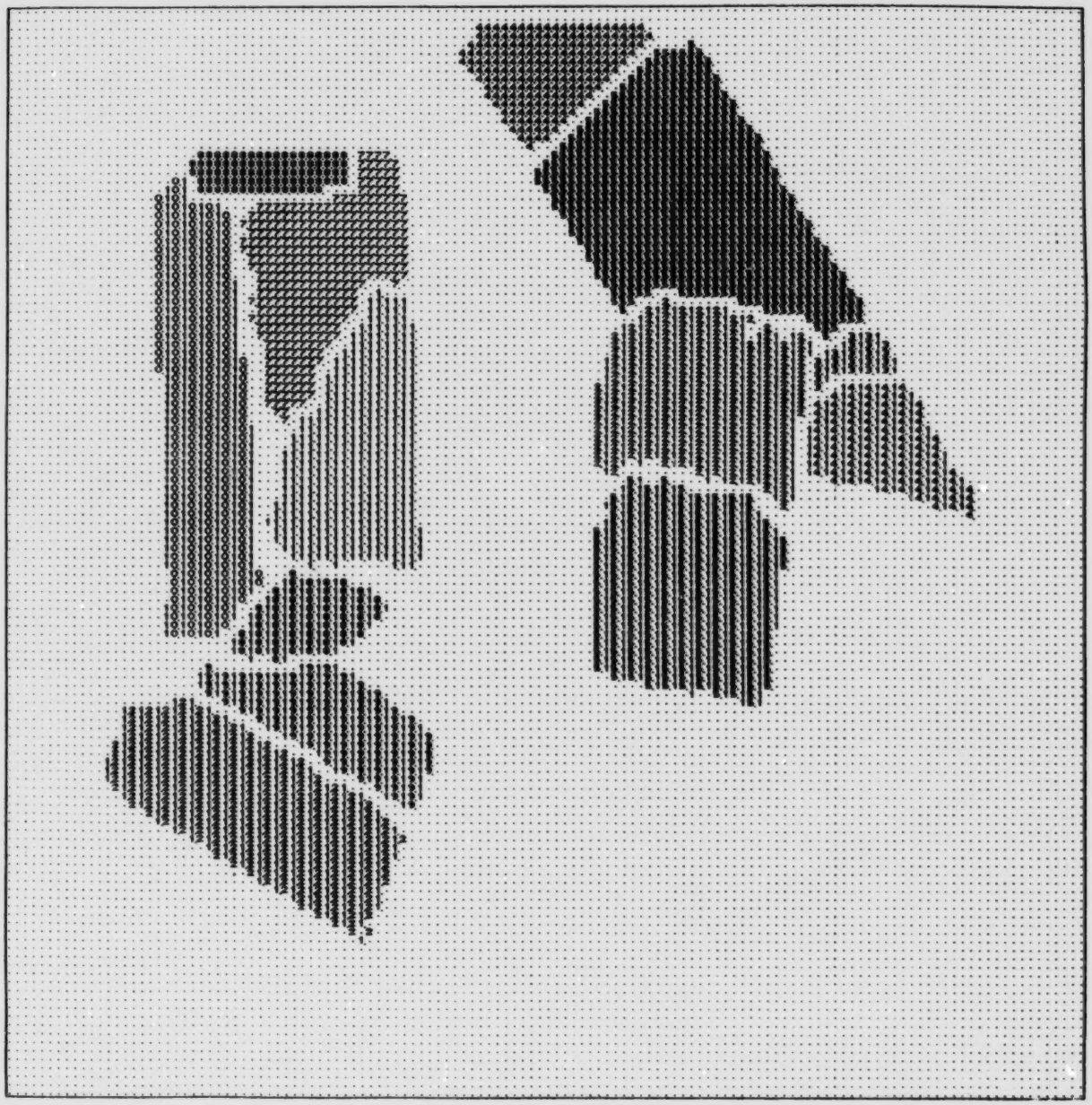
(a)

Figure 9. (a) Range data. (b) Visualization of normals at each point of the scene. (c) Non-vertical planar surfaces extracted from the range data. (d) Overlapping of normal images and selected models.



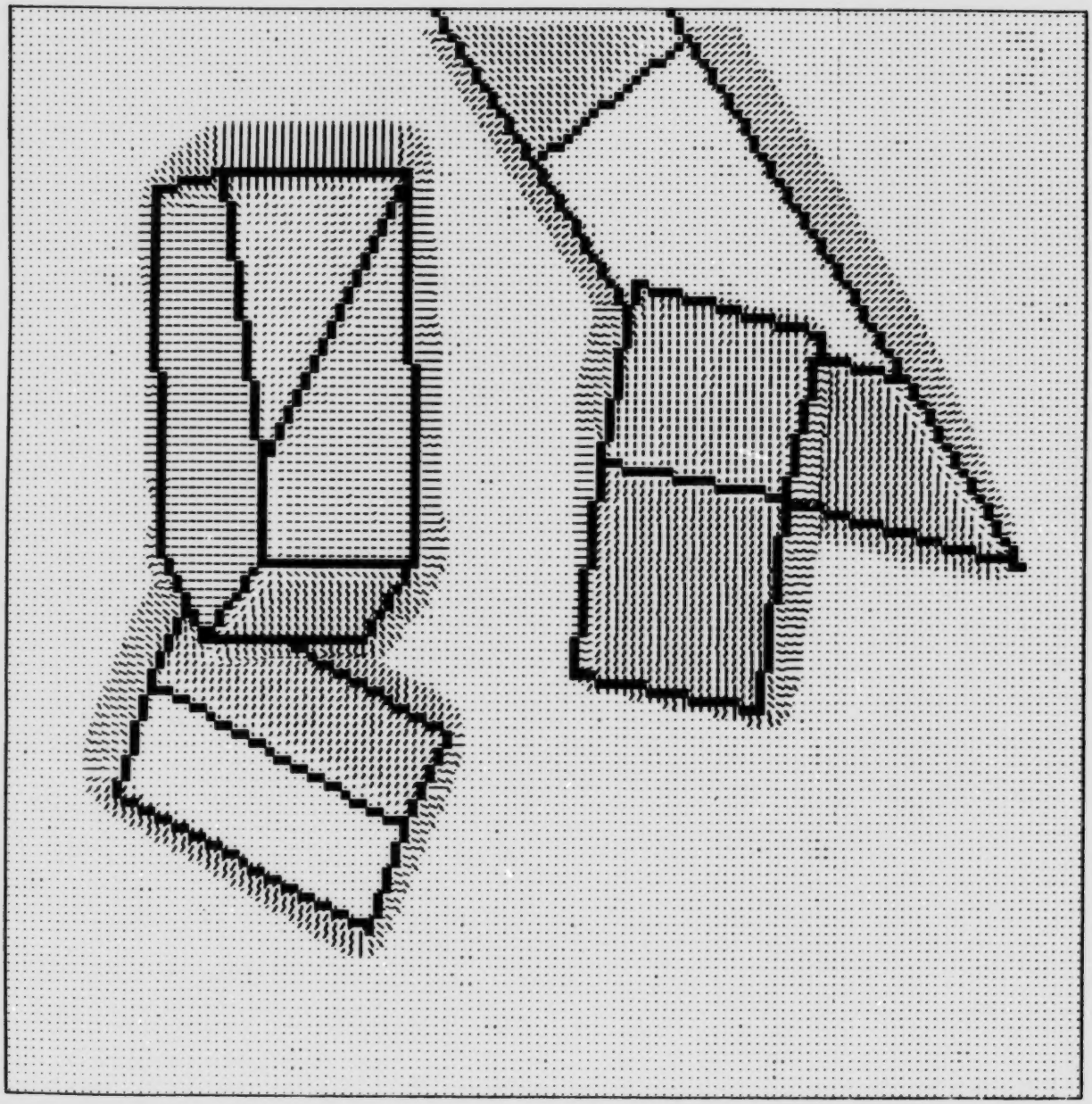
(b)

Figure 9. (Continued).



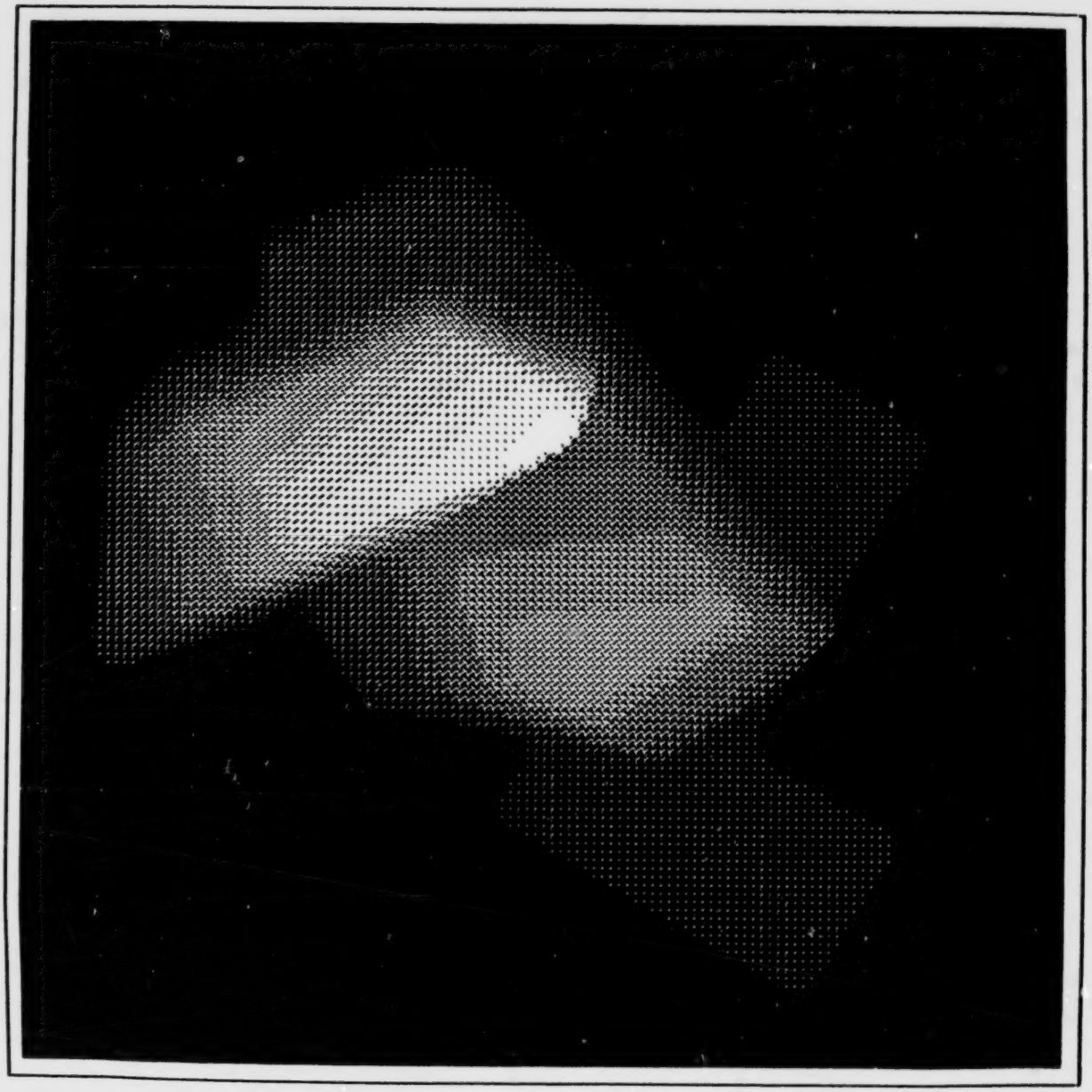
(c)

Figure 9. (Continued).



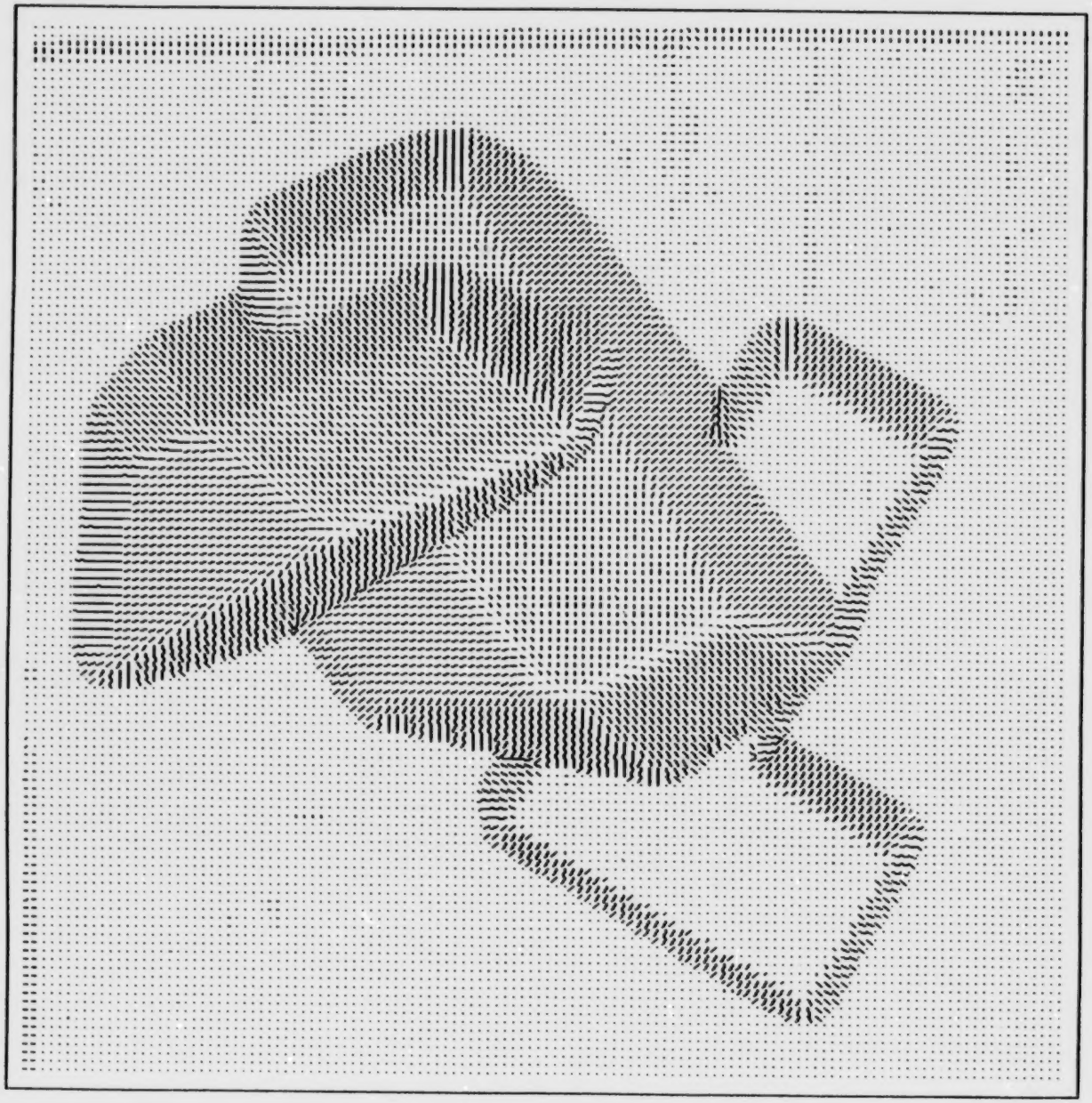
(d)

Figure 9. (Concluded).



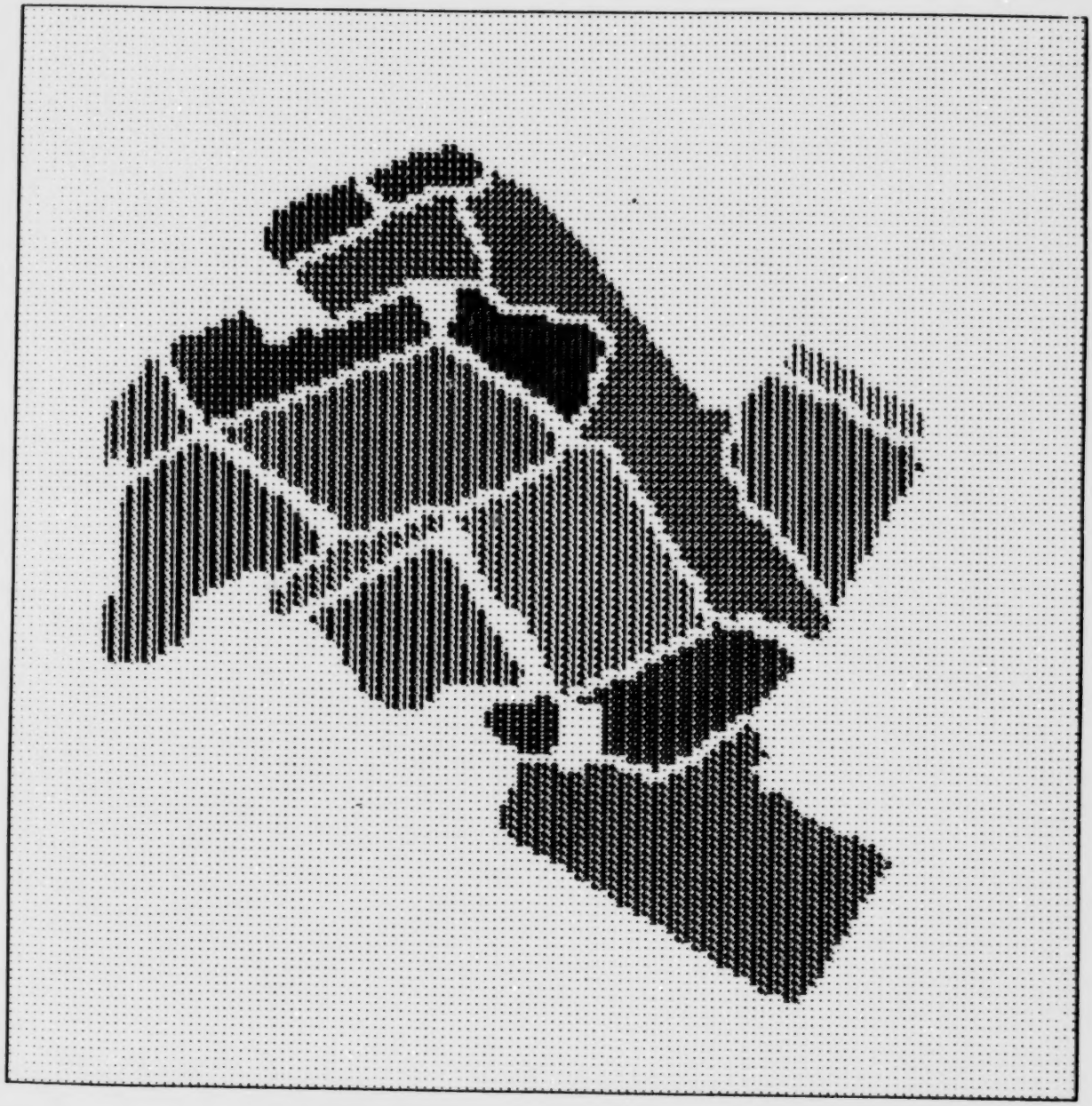
(a)

Figure 10. (a) Range data. (b) Visualization of normals at each point of the scene. (c) Non-vertical planar surfaces extracted from the range data. (d) Overlapping of normal image and selected models.



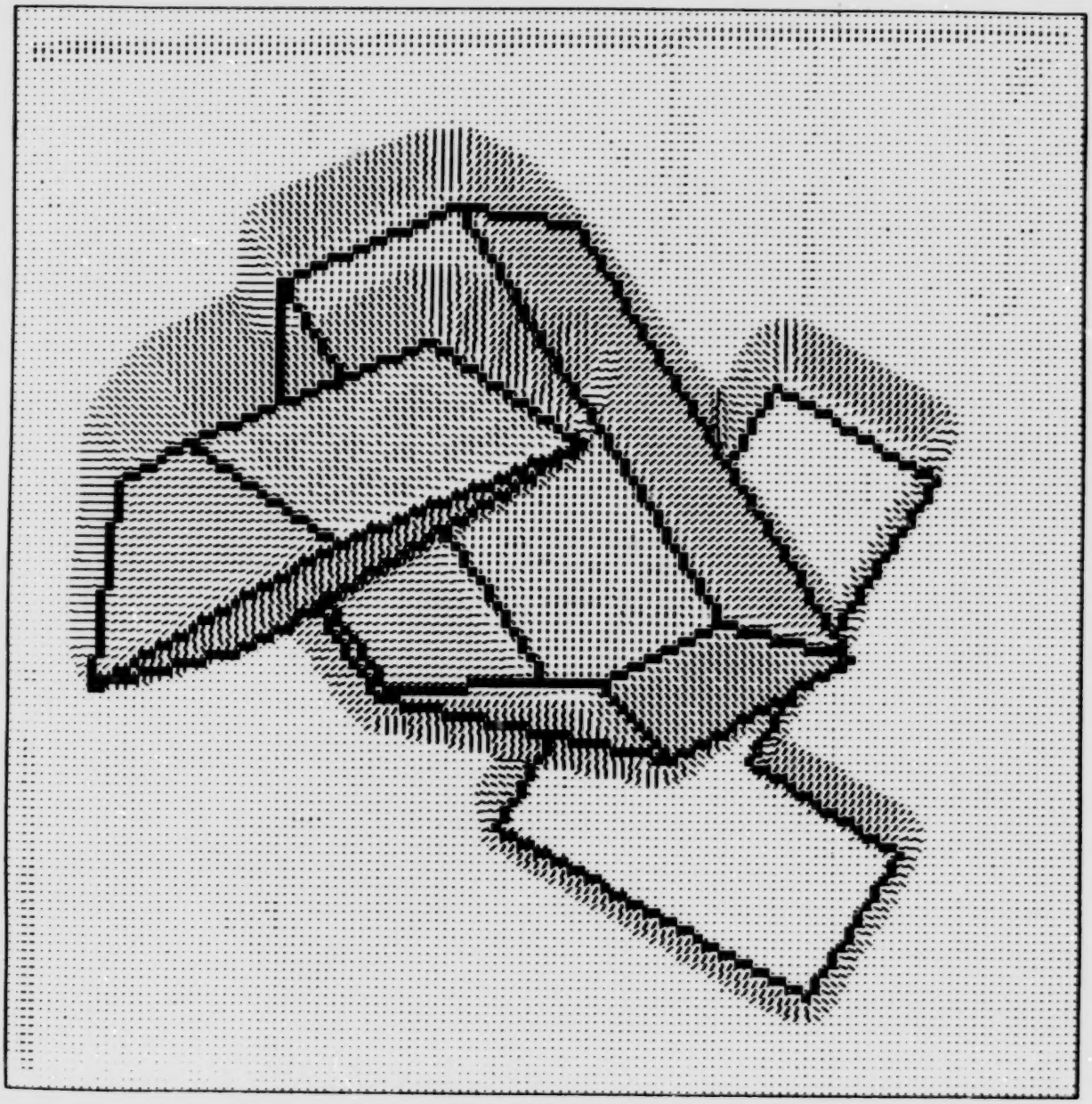
(b)

Figure 10. (Continued).



(c)

Figure 10. (Continued).



(d)

Figure 10. (Concluded).

Table 3. Results of the analysis of the scene presented in Fig. 10.

	Object	Model	Rotation (°)		Translation (mm)		Index (mm)
Top	1	1	R_x	98	T_x	56	0.90
			R_y	-32	T_y	30	
			R_z	-72	T_z	41	
	3	3	R_x	26	T_x	56	3.70
			R_y	4	T_y	52	
			R_z	-57	T_z	27	
	7	7	R_x	0	T_x	82	1.01
			R_y	0	T_y	70	
Bottom			R_z	59	T_z	11	

IV. CONCLUSION

The algorithm of polyhedra recognition in range data proposed in this paper is based on a principle of hypothesis accumulation. The models of the objects are described in terms of local geometric patterns (adjacent surfaces). From the studied scene the same elements are extracted. Each matching between scene and model patterns allows the generation of a hypothesis about the position of an object in the scene similar to the studied model. The recognition process is based on the detection of large sets of agreeing hypotheses. For that, the method does not require an accumulator array but uses a special hierarchical scheme working on three parameters defining the position of the objects in the scene.

The experimental results prove that this method can correctly analyse artificial or real scenes with overlapping objects (selection of the good models for each objects in the scene and precise determination of their positions).

For industrial applications, the recognition of objects by a robot vision system must be as fast as possible. The proposed algorithm has an important advantage with regards to this constraint because it can be executed with a parallel computer structure since the treatements at many different levels are completely independent.

The future direction of our work is to insert additional local geometrical patterns, such as spherical, cylindrical, and conical surfaces, in the LGP-Model with the goal of analysing scenes containing more complex objects.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to the director of the Division of the Electrical Engineering, Dr. J.Y. Wong, the section head, Mr. N. Burtnyk, and the staff of the Computer Graphics Section for making this study possible. Thanks are also due to the staff of the Electronic Engineering Section for providing all the range data images needed for the experiments.

REFERENCES

1. N. Ayache and O.D. Faugeras. A new method for the recognition and the positioning of 2-D objects. Proc. of 7th Int. Conf. on Pattern Recognition, Montreal, P.Q. July 1984. pp. 1274-1277.

2. D.H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. Pattern Recog. 13(2): 111-122; 1981.

3. R.C. Bolles and R.A. Cain. Recognizing and locating partially visible work-pieces. Proc. of Pattern Recognition in Practice, Las Vegas, NV. July 1982. pp. 498-503.

4. M. Dhome, M. Richetin, and G. Rives. Model-based recognition and location of local patterns in polygonal contours via hypothesis accumulation. Proc. of Pattern Recognition in Practice II, Amsterdam, The Netherlands. June 1985.

5. T.J. Fang, Z.H. Huang, L.N. Kanal, B. Lambird, D. Lavine, G. Stockman, and F.L. Xiong. Three dimensional object recognition using a transformation clustering technique. Proc. of 6th Int. Conf. on Pattern Recognition, Munich. October 1982. pp. 678-681.

6. H. Hakalahti, D. Harwood, and L.S. Davis. Two-dimensional object recognition by matching local properties of contour points. Pattern Recog. Lett. 2: 227-234; 1984.

7. T.M. Silberberg, L. Davis and H. Harwood. An iterative Hough procedure for three-dimensional object recognition. Pattern Recog. 17(6): 621-620; 1984.

8. M. Rioux. Laser range finder based on synchronized scanners. Appl. Opt. 23(21): 3837-3844; 1984.

9. M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. To be published.

10. M. Hebert and J. Ponce. A new method for segmenting 3-D scenes into primitives. Proc. of 6th Int. Conf. on Pattern Recognition, Munich, October 1982. pp. 836-838.

11. T.C. Henderson and B. Bhanu. Three-point seed method for extraction of planar faces from range data. Proc. of IEEE Workshop on Industrial Applications of Machine Vision, Research Triangle Park, North California, CA. May 1982. pp. 181-186.

12. D.J. Ittner and A.K. Jain. 3-D surface discrimination from local curvature measures. Proc. of Comp. Vision and Pattern Recognition, San Francisco, CA. June 1985. pp. 119-123.

13. T. Kasvand, R. Oka, M. Dhome, and M. Rioux. Segmentation, learning, and recognition of objects in range camera images. 4th Canadian CAD/CAM and Robotics Conference, Toronto, Ont. June 1985.

14. R. Bajcsy. Three-dimensional scene analysis. Proc. of 5th Int. Conf. on Pattern Recognition, Miami Beach, FL. December 1980. pp. 1064-1074.

15. O.D. Faugeras and M. Hebert. A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. Proc. of 8th Int. Joint Conf. on Artificial Intelligence, Los Altos, 1983. pp. 996-1002.

16. P. Horaud and R.C. Bolles. 3DPO's strategy for matching three-dimensional objects in range data. Proc. of Int. Conf. on Robotics, Atlanta, GA. March 1984. pp. 78-85.

17. R. Oka, T. Kasvand, and M. Rioux. Cross-angle transform for viewer-independent recognition of 3-D objects. Proc. of Comp. Vision and Pattern Recognition, San-Francisco, CA. June 1985. pp. 470-475.

18. M. Oshime, and Y. Shirai. Object recognition using three-dimensional information. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5(4): 353-361; 1983.

19. K. Sugihara. Range-data analysis guided by a junction dictionary. Artificial Intell. 12: 41-69; 1979.

20. H.T. Tanaka, D.H. Ballard, S. Tsuji, and M. Curtiss. Parallel polyhedral shape recognition. Proc. of Comp. Vision and Pattern Recognition, San-Francisco, June 1985, pp 491-496.

21. Wong, A.K.C., and Lu, S.W., Recognition and knowledge synthesis of 3-D object images. Proc. of Comp. Vision and Pattern Recognition Conference, San Francisco, CA. June 1985. pp. 162-166.

22. W.A.C. Perkins. A model-based vision system for industrial parts. IEEE Trans. Comput. C-27(2): 126-143; 1978.

23. Hough, P.V.C. Method and means for recognizing complex pattern. U.S. Patent 3069654. 1962.

24. T.C. Henderson and W.S. Fai. The 3-D Hough shape transform. Pattern Recog. Lett. 2: 235-238; 1984.

25. O.R. Duda and P.E. Hart. Pattern Classification and Scene Analysis. John Wiley and Sons, New-York, NY. 1973.

APPENDIX

The readers will find in the following pages a succinct description of the goal, the parameters, the input and output files and the listing of each program used during the process of the Polyhedra Recognition. These programs are written in PASCAL.

These programs can be classified according to their goals in five groups:

- Modelling of the Polyhedra
  - "CreationObjet" program
  - "LectureObjet" program
  - "VisualisationObjet" program
  - "ModelisationObjet" program
- Creation Artificial Scenes
  - "CreationImages" program
- Extraction of Characteristics of the Studied Scene
  - "LectureScene" program
- Recognition of Polyhedra
  - "AnalyseScene" program
- Visualisation of the different Results
  - "ImprimeImage" program
  - "ProjectionObjet" program

A. "CREATIONOBJET" PROGRAM

This program is used to create the "CAD-Model" of a polyhedron. For more informations about the definition of the "CAD-Model" see paragraph II.1.

Parameters

- Name of the file containing the CAD-Model. This name must agree with the following format [prefix + number . extension] (e.g., MODEL12.CAD). The name of the CAD-Models used during an experiment must have the same "prefix", their "numbers" must be given in a continuous way varying from 1 to N (N is equal to the number of different models) and theirs "extensions" must be equal to ".CAD".
- Number of vertices of the polyhedron.
- Coordinates of each vertice of the polyhedron.
- Number of faces of the polyhedon.
- Index of the vertices belonging to each face.
- Index of the adjacent faces to each face.

Output File

- File containing the CAD-Model of the polyhedron.

```
{*****************************************************************************}
                PROGRAM CreationObjet(INPUT,OUTPUT,FicObjet);
{*****************************************************************************}

{======================== DECLARATION DE TYPES ============================}
= TYPE
  Point = RECORD
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Surface = RECORD
        Nor      :  Vecteur;
        Cen      :  Point;
        Taille   :  INTEGER;
        NbSomLie :  1..20;
        SomLie   :  ARRAY[1..20] OF 1..100;
        NbSurAdj :  1..20;
        SurAdj   :  ARRAY[1..20] OF 1..20;
        END;
  Objet = RECORD
        NbSom : 1..100;
        Som   : ARRAY[1..100] OF Point;
        NbSur : 1..20;
        Sur   : ARRAY[1..20] OF Surface;
        END;


{======================= DECLARATION DE VARIABLES ==========================}

VAR
  Obj              : Objet;
  IndSur,IndSurAdj : 1..20;
  IndSom,IndSomLie : 0..100;
  Nomfic           : PACKED ARRAY[1..12] OF CHAR;
  FicObjet         : FILE OF Objet;
  V1,V2,V3         : Vecteur;
  Choix            : VARYING[1] OF CHAR;
  Pc               : Point;
  SomTaille        : Integer;
```

```
{======================= DECLARATION DE PROCEDURES ======================}

PROCEDURE ProdVect(Vi,Vj : Vecteur; VAR Vk : Vecteur);              EXTERN;
PROCEDURE ProdVectNor(Vi,Vj : Vecteur; VAR Vk : Vecteur);           EXTERN;
FUNCTION  ProdScal(V1,V2 : Vecteur) : REAL;                         EXTERN;


{*************************************************************************}
{*                            PROGRAMME PRINCIPAL                        *}
{*************************************************************************}

BEGIN
WRITELN('                    **** DEFINITION D''OBJETS ****');
REPEAT WRITELN;WRITELN;WRITELN;
  WRITE(' Nom du fichier objet ? ');READLN(NomFic);OPEN(FicObjet,NomFic);
  WRITELN;WRITELN;WRITELN;
  WITH Obj DO
    BEGIN
    WRITE(' Nombre de Sommets ? ');READLN(NbSom);WRITELN;WRITELN;
    FOR IndSom := 1 TO NbSom DO
      BEGIN WRITE(' coordonnees du ',IndSom:3,' sommet ? ');
      WITH Som[IndSom] DO READLN(x,y,z);
      END;
    WRITELN;WRITELN;WRITELN;
    WRITE(' Nombre de Surfaces ? ');READLN(NbSur);WRITELN;WRITELN;
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur] DO
        BEGIN Cen.x:=0; Cen.y:=0; Cen.z:=0;IndSomlie:=0;IndSurAdj:=0;
        WRITE(' Indices des sommets de la ',IndSur:2,' surface ? ');
        REPEAT
          IndSomLie:=IndSomLie+1; READ(Somlie[IndSomLie]);
          WITH Som[SomLie[IndSomLie]] DO
            BEGIN Cen.x:=Cen.x+x; Cen.y:=Cen.y+y; Cen.z:=Cen.z+z; END;
        UNTIL EOLN(INPUT);
        NbSomLie:=IndSomLie; READLN;
        WRITE(' Indices des surfaces adjacentes a la ',IndSur:2,
                                                        ' surface ? ');
        REPEAT
          IndSurAdj:=IndSurAdj+1; READ(SurAdj[IndSurAdj]);
        UNTIL EOLN(INPUT);
        NbSurAdj:=IndSurAdj; READLN;WRITELN;
        WRITE(' Calcul automatique du centre ? ');READLN(Choix);
        WITH Cen DO
          IF (LENGTH(Choix)=0)OR(Choix='o')OR(Choix='O') THEN
            BEGIN x:=ROUND(x/NbSomLie); y:=ROUND(y/NbSomLie);
            z:=ROUND(z/NbSomLie);
            WRITELN(' Coordonnees du centre : ',x:4,Y:4,z:4);
```

```
            END
          ELSE
            IF (Choix='N')OR(Choix='n') THEN
              BEGIN WRITE(' Coordonnees du centre ? ');READLN(x,y,z); END;
        WRITELN;WRITELN;
        WITH Som[Somlie[2]],V1 DO
          BEGIN
          u:=Som[SomLie[1]].x-x;v:=Som[SomLie[1]].y-y;w:=Som[Somlie[1]].z-z;
          END;
        WITH Som[SomLie[2]],V2 DO
          BEGIN
          u:=Som[SomLie[3]].x-x;v:=Som[SomLie[3]].y-y;w:=Som[SomLie[3]].z-z;
          END;
        ProdVectNor(V1,V2,Nor);Taille:=0;
        FOR IndSomLie :=1 TO NbSomLie DO
          BEGIN
          WITH Som[SomLie[IndSomlie]],V1 DO
            BEGIN u:=x-Cen.x; v:=y-Cen.y; w:=z-Cen.z; END;
          IF IndSomlie<NbSomLie THEN
            WITH Som[SomLie[IndSomlie+1]],V2 DO
              BEGIN u:=x-Cen.x; v:=y-Cen.y; w:=z-Cen.z; END
          ELSE
            WITH Som[SomLie[1]],V2 DO
              BEGIN u:=x-Cen.x; v:=y-Cen.y; w:=z-Cen.z; END;
          ProdVect(V2,V1,V3); Taille:=Taille+ROUND(ProdScal(V3,Nor));
          END;
        Taille:=ROUND(Taille/2);
        END;
      WITH Pc DO
        BEGIN
        FOR IndSur:=1 TO NbSur DO
          WITH Sur[IndSur] DO
            BEGIN x:=x+Cen.x*Taille; y:=y+Cen.y*Taille; z:=z+Cen.z*taille;
            SomTaille:=SomTaille+Taille;
            END;
        x:=ROUND(x/SomTaille); y:=ROUND(y/SomTaille); z:=ROUND(z/SomTaille);
        END;
      FOR IndSur:=1 TO NbSur DO
        WITH Sur[IndSur],Cen DO BEGIN x:=x-Pc.x; y:=y-Pc.y; z:=z-Pc.z; END;
      FOR IndSom:=1 TO NbSom DO
        WITH Som[IndSom] DO BEGIN x:=x-Pc.x; y:=y-Pc.y; z:=z-Pc.z; END;
      END;
    WRITELN;WRITELN;REWRITE(FicObjet);WRITE(FicObjet,Obj);CLOSE(FicObjet);
    WRITE(' autre objet a definir ? ');READLN(Choix);
  UNTIL (Choix='n') OR (Choix='N');
  END.
```

B. "LECTUREOBJET" PROGRAM

This program is used to read the content of the CAD-Model of the polyhedron and to write it in a text file.

Parameters

- Name of the CAD-Model of the polyhedron.
- Name of the text file.

Input File

- File containing the CAD-Model of the polyhedron.

Output File

- Text file corresponding to the CAD-Model of the polyhedron.
  Figure A-1 is an example of a text file corresponding to the object No. 2 presented in Fig. 8a.

```
FICHIER : MODEL2.CAD

Nombre de Sommets :   8

Coordonnees  1 sommet :    0   0   0
Coordonnees  2 sommet :   12   0   0
Coordonnees  3 sommet :   12  18   0
Coordonnees  4 sommet :    0  18   0
Coordonnees  5 sommet :    0   0  12
Coordonnees  6 sommet :   12   0  12
Coordonnees  7 sommet :   12   5  12
Coordonnees  8 sommet :    0   5  12

Nombre de surfaces :    6

---------------- 1 Surface -------------------
Normale.............. :  0.0E+00 0.0E+00-1.0E+00
Centre............... :     6    9    0
Surface.............. :       216
Sommets Lies......... :   1  2  3  4
Surfaces adjacentes.. :   5  2  6  4
---------------- 2 Surface -------------------
Normale.............. :  1.0E+00 0.0E+00 0.0E+00
Centre............... :    12    6    6
Surface.............. :       138
Sommets Lies......... :   2  6  7  3
Surfaces adjacentes.. :   5  3  6  1
---------------- 3 Surface -------------------
Normale.............. :  0.0E+00 0.0E+00 1.0E+00
Centre............... :     6    3   12
Surface.............. :        60
Sommets Lies......... :   5  8  7  6
Surfaces adjacentes.. :   4  6  2  5
---------------- 4 Surface -------------------
Normale.............. : -1.0E+00 0.0E+00 0.0E+00
Centre............... :     0    6    6
Surface.............. :       138
Sommets Lies......... :   1  4  8  5
Surfaces adjacentes.. :   1  6  3  5
---------------- 5 Surface -------------------
Normale.............. :  0.0E+00-1.0E+00 0.0E+00
Centre............... :     6    0    6
Surface.............. :       144
Sommets Lies......... :   1  5  6  2
Surfaces adjacentes.. :   4  3  2  1
---------------- 6 Surface -------------------
Normale.............. :  0.0E+00 6.8E-01 7.3E-01
Centre............... :     6   12    6
Surface.............. :       213
Sommets Lies......... :   3  7  8  4
Surfaces adjacentes.. :   2  3  4  1
```

```
{*************************************************************************}
                    PROGRAM LectureObjet(INPUT,OUTPUT);
{*************************************************************************}


{======================== DECLARATION DE TYPES ========================}

TYPE
  Point = RECORD
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Surface = RECORD
        Nor      : Vecteur;
        Cen      : Point;
        Taille   : INTEGER;
        NbSomLie : 1..20;
        SomLie   : ARRAY[1..20] OF 1..100;
        NbSurAdj : 1..20;
        SurAdj   : ARRAY[1..20] OF 1..20;
        END;
  Objet = RECORD
        NbSom : 1..100;
        Som   : ARRAY[1..100] OF Point;
        NbSur : 1..20;
        Sur   : ARRAY[1..20] OF Surface;
        END;


{====================== DECLARATION DE VARIABLES ======================}

VAR
  Obj                    : Objet;
  IndSur,IndSurAdj       : 1..20;
  IndSom,IndSomLie       : 1..100;
  NomficObj,NomFicLecObj : PACKED ARRAY[1..12] OF CHAR;
  FicObjet               : FILE OF Objet;
  FicLecObjet            : TEXT;
  Choix                  : CHAR;
  Ori                    : Point;
```

```
{*****************************************************************************}
{*                          PROGRAMME PRINCIPAL                              *}
{*****************************************************************************}

BEGIN
WRITELN('                        **** LECTURE FICHIER OBJET ****');
REPEAT
  WRITELN;WRITELN;WRITELN;
  WRITE(' Nom du fichier Objet ? ');READLN(NomFicObj);
  OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
  CLOSE(FicObjet);WRITELN;
  WRITE(' Nom du fichier texte ? ');READLN(NomFicLecObj);
  OPEN(FicLecObjet,NomFicLecObj);REWRITE(FicLecObjet);
  WRITELN;WRITELN;
  FOR IndSom:=1 TO 80 DO WRITE(FicLecObjet,'*');WRITELN(FicLecObjet);
  WRITELN(FicLecObjet,' ':25,'FICHIER OBJET : ',NomFicObj);
  FOR IndSom:=1 TO 80 DO WRITE(FicLecObjet,'*');WRITELN(FicLecObjet);
  WRITELN(FicLecObjet);
  WITH Obj DO
    BEGIN WRITELN(FicLecObjet,' Nombre de Sommets : ',NbSom:3);
    WRITELN(FicLecObjet); Ori:=Som[1];
    FOR IndSom:=1 TO NbSom DO
      WITH Som[IndSom] DO BEGIN x:=x-Ori.x; y:=y-Ori.y; z:=z-Ori.z; END;
    FOR IndSom:=1 TO NbSom DO
      WITH Som[IndSom] DO
        WRITELN(FicLecObjet,' Coordonnees ',IndSom:2,' sommet : ',
                                                    X:4,Y:4,Z:4);
    WRITELN(FicLecObjet);WRITELN(FicLecObjet);
    WRITELN(FicLecObjet,' Nombre de surfaces : ',NbSur:4);
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur] DO
        BEGIN
        WRITELN(FicLecObjet);WRITELN(FicLecObjet,'  ',IndSur:2,' Surface');
        WRITELN(FicLecObjet,' Normale..... : ',
                                        Nor.u:5,Nor.v:5,Nor.w:5);
        WRITELN(FicLecObjet,' Centre..... : ',Cen.x-Ori.x:5,
                                  Cen.y-Ori.y:5,Cen.z-Ori.z:5);
        WRITELN(FicLecObjet,' Surface..... : ',Taille:8);
        WRITE(FicLecObjet,' Sommets Lies..... : ');
        FOR IndSomLie:=1 TO NbSomLie DO WRITE(FicLecObjet,
                                        Somlie[IndSomlie]:3);
        WRITELN(FicLecObjet);
        WRITE(FicLecObjet,' Surfaces adjacentes.. : ');
        FOR IndSurAdj:=1 TO NbSurAdj DO WRITE(FicLecObjet,
                                        SurAdj[IndSurAdj]:3);
```

```
        WRITELN(FicLecObjet);
        END;
    END;
  CLOSE(FicLecObjet);
  WRITE(' Autre fichier a lire ? ');READLN(Choix);
UNTIL (Choix='n') OR (Choix='N');
END.
```

C. "VISUALISATIONOBJET" PROGRAM

This program allows to create a visual representation of the polyhedra from their CAD-Models (see figure No 12). The result of this program is a text file. The point of view of these polyhedra is fixed by the user who gives the values of rotations and translations to apply to the CAD-Models and the values of scales on the three axis of the space.

Parameters

- Name of the text file corresponding to the visual representation.
- Millimeter distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis) and the between two levels on the elevation image axis (Z-axis).
- Name of the file containing the CAD-Models of the polyhedra.
- Values of the translations to be applied to the CAD-Models.
- Values of the rotations to be applied to the CAD-Models.

Input Files

- Files containing the CAD-Models of the polyhedra.

Output File

- Text file corresponding to a visual representation of the polyhedra.

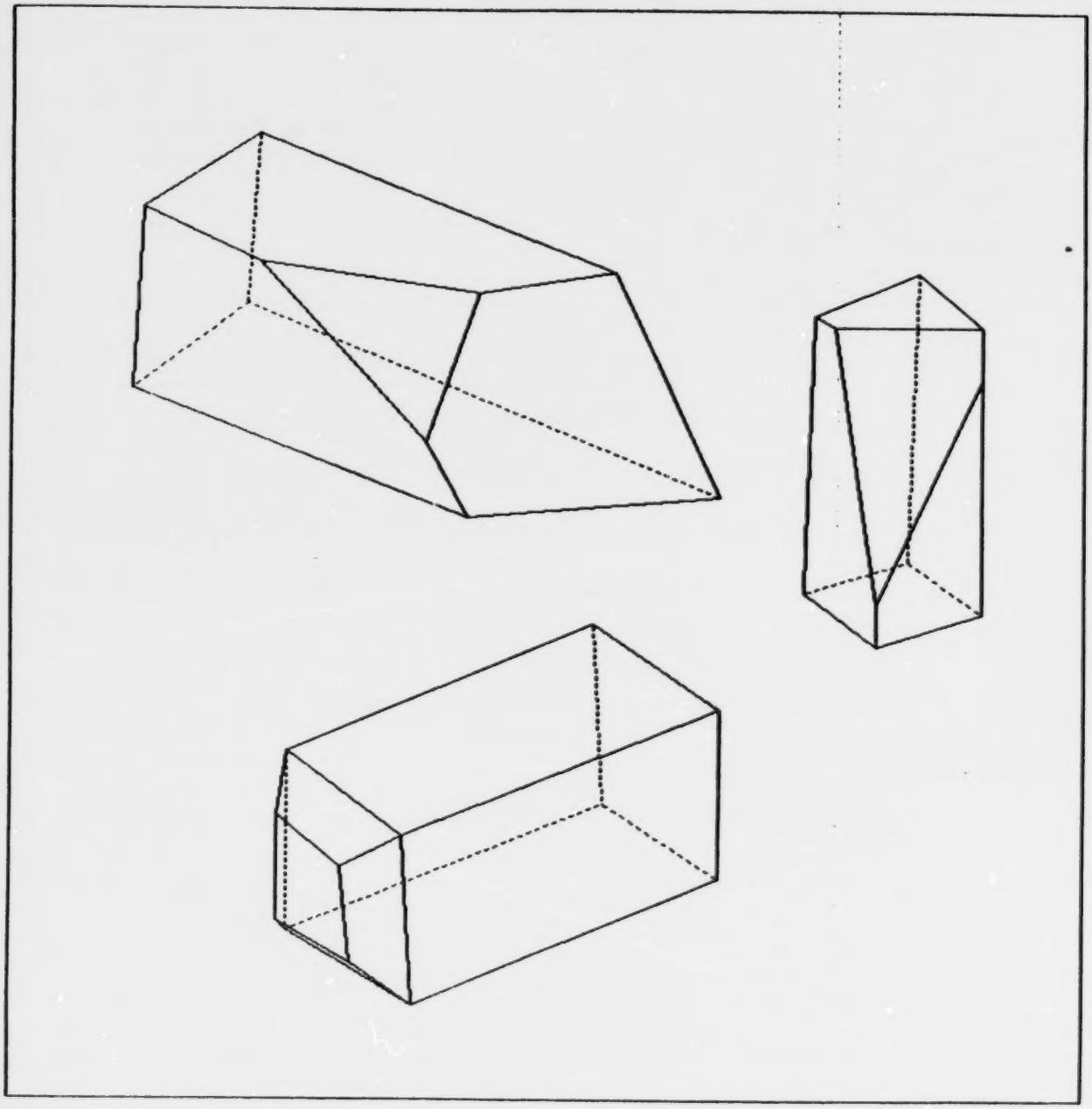


Figure A-2. Visualisation of the objects No. 4,10,11 presented in Fig. 8a.

```
{*****************************************************************************}
                  PROGRAM VisualisationObjet(INPUT,OUTPUT);
{*****************************************************************************}


{========================= DECLARATION DE TYPES ============================}

TYPE
  Degre = -180..180;
  Point = RECORD
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Rotation = RECORD
        a,b,g : INTEGER;
        END;
  Surface = RECORD
        Nor : Vecteur;
        Cen : Point;
        Taille : INTEGER;
        NbSomLie : 1..20;
        SomLie : ARRAY[1..20] OF 0..100;
        NbSurAdj : 1..20;
        SurAdj : ARRAY[1..20] OF 0..20;
        END;
  Objet = RECORD
        NbSom : 1..100;
        Som : ARRAY[1..100] OF Point;
        NbSur : 1..20;
        Sur : ARRAY[1..20] OF Surface;
        END;
  CONV = RECORD CASE BOOLEAN OF
        TRUE :(TEX:ARRAY[1..107] OF CHAR);
        FALSE :(BOO:PACKED ARRAY[1..856] OF BOOLEAN);
        END;


{====================== DECLARATION DE VARIABLES ==========================}

VAR
  NomficObj,Tampon,NomFicText : PACKED ARRAY[1..12] OF CHAR;
  FicObjet                    : FILE OF Objet;
  FicText                     : TEXT;
```

```
  A                              : PACKED ARRAY[1..768,1..642] OF BOOLEAN;
  B                              : CONV;
  Obj                            : Objet;
  IndSur,IndSurAdj,IndSurAdj2    : 1..10;
  IndSom,IndSomLie               : 1..100;
  Tra                            : Vecteur;

  Rot                            : Rotation;
  P1,P2                          : Point;
  CoefX,CoefY                    : REAL;
  Choix                          : CHAR;


{======================= DECLARATION DE PROCEDURES ======================}

PROCEDURE RotationVect(Rot : Rotation; VAR Vec : Vecteur);          EXTERN;
PROCEDURE RotationPoint(Rot : Rotation; VAR P : Point);            EXTERN;
PROCEDURE Translation(Ttr : Vecteur; VAR P : Point);               EXTERN;

{=====================================================================}
                          PROCEDURE InitTabBool;
{=====================================================================}

VAR
  I,J : INTEGER;

BEGIN
FOR J:=1 TO 642 DO A[1,J]:=FALSE;
FOR J:=2 TO 768 DO A[J]:=A[1];
FOR I:=1 TO 768 DO BEGIN A[I,1]:=TRUE; A[I,640]:=TRUE; END;
FOR J:=1 TO 640 DO BEGIN A[1,J]:=TRUE; A[768,J]:=TRUE; END;
END;


{=====================================================================}
          PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER);
{=====================================================================}

VAR
  Xi,Xs,Yi,Ys : INTEGER;
  I,J : INTEGER;
  Dx,Dy,Pente,Constante : REAL;

BEGIN
WITH P1 DO BEGIN x:=ROUND((x*6+4)/CoefX); y:=ROUND((y*5+3)/CoefY); END;
```

```
WITH P2 DO BEGIN x:=ROUND((x*6+4)/CoefX); y:=ROUND((y*5+3)/CoefY); END;
WITH P1 DO
  BEGIN Dx:=x-P2.x; Dy:=y-P2.y;
  IF (Dx=0) AND (Dy=0) THEN A[x,y]:=TRUE
  ELSE
    IF ABS(Dx)>ABS(Dy) THEN
      BEGIN
      IF x<P2.x THEN BEGIN Xi:=x;Xs:=P2.x; END ELSE BEGIN Xi:=P2.x;Xs:=x; END;
      Pente:=Dy/Dx; Constante:=y-Pente*x;
      FOR I:=Xi TO Xs DO
        BEGIN J:=ROUND(Pente*I+Constante);
        IF (J>=1)AND(J<=640)AND(I>=1)AND(I<=768) THEN
          BEGIN
          IF Poids=1 THEN A[I,J]:=TRUE
          ELSE IF (I MOD 5=0)OR(I MOD 5=1)OR(I MOD 5=2) THEN A[I,J]:=TRUE;
          END;
        END;
      END
    ELSE
      BEGIN
      IF y<P2.y THEN BEGIN Yi:=y;Ys:=P2.y; END ELSE BEGIN Yi:=P2.y;Ys:=y; END;
      Pente:=Dx/Dy; Constante:=x-Pente*y;
      FOR J:=Yi TO Ys DO
        BEGIN I:=ROUND(Pente*J+Constante);
        IF (I>=1)AND(I<=768)AND(J>=1)AND(J<=640) THEN
          BEGIN
          IF Poids=1 THEN A[I,J]:=TRUE
          ELSE IF (J MOD 5=0)OR(J MOD 5=1)OR(J MOD 5=2) THEN A[I,J]:=TRUE;
          END;
        END;
      END;
  END;
END;


{=============================================================================}
                              PROCEDURE ImpTabBool;
{=============================================================================}

VAR
  I,J,K : INTEGER;

BEGIN
FOR I:=1 TO 65 DO WRITELN(FicText);
FOR I:=1 TO 768 DO
```

```
  BEGIN
  FOR J:=1 TO 107 DO
    FOR K:=1 TO 8 DO
      CASE K OF
        7 : B.BOO[(J-1)*8+7]:=TRUE;
        8 : B.BOO[(J-1)*8+8]:=FALSE;
        OTHERWISE B.BOO[(J-1)*8+K]:=A[I,(J-1)*6+K];
      END;
  WRITE(FicText,CHR(5));
  FOR J:=1 TO 107 DO
  WRITE(FicText,B.TEX[J]);
  WRITELN(FicText);
  END;
END;


{*****************************************************************************}
{*PROGRAMME PRINCIPAL*}
{*****************************************************************************}


BEGIN WRITELN('**** VISUALISATION D''UN OBJET ****');
WRITELN;WRITELN;WRITELN;
WRITE(' Nom du fichier text ? ');READLN(NomFicText);
InitTabBool;
REPEAT
  WRITELN;WRITELN;
  WRITE(' Nom du fichier Objet ? ');READLN(NomFicObj);
  IF NomFicObj[1]=CHR(32) THEN NomFicObj:=Tampon
  ELSE Tampon:=NomFicObj;
  OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
  CLOSE(FicObjet);
  WRITE(' Parametres de la translation ? ');WITH Tra DO READLN(u,v,w);
  WRITE(' Parametres de la rotation ? ');WITH Rot DO READLN(a,b,g);
  WRITE(' Valeur Coefficients en X,Y ? ');READLN(CoefX,CoefY);
  WITH Obj DO
    BEGIN
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur] DO RotationVect(Rot,Nor);
    FOR IndSom := 1 TO NbSom DO
      WITH Som[IndSom] DO
        BEGIN RotationPoint(Rot,Som[IndSom]);Translation(Tra,Som[IndSom]);END;
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur] DO
        FOR IndSomLie:= 1 TO NbSomLie DO
```

```
          BEGIN P1:=Som[Somlie[IndSomlie]];
          IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
          ELSE P2:=Som[Somlie[IndSomlie+1]];
          IF Nor.w>=0 THEN ProjecSegment(P1,P2,1) ELSE ProjecSegment(P1,P2,2);
          END;
    END;
  WRITE(' Autre projection ? ');READLN(CHOIX);
UNTIL (Choix='n')OR(CHOIX='N');
OPEN(FicText,NomFicText,NEW,270); REWRITE(FicText); ImpTabBool;
END.
```

D. "MODELISATIONOBJET" PROGRAM

This program is used to create the "LGP-Model" of the polyhedron. For more informations about the definition of the LGP-Model see paragraph II.1.

Parameters

- Name of the CAD-Model of the polyhedron.

- Name of the LGP-Model of the polyhedron. This name must agree with the following format: [prefix + number . extension] (e.g., MODEL12.LGP). The first part (name + number) of this name must be equal to the corresponding CAD-Model one and its extension must be equal to ".LGP".

Input File

- File containing the CAD-Model of the Polyhedron.

Output File

- File containing the LGP-Model of the polyhedron.

```
{*****************************************************************************}
                    PROGRAM ModelisationObjet(INPUT,OUTPUT);
{*****************************************************************************}


{========================= DECLARATION DE TYPES ============================}

TYPE
  Degre = -180..180;
  Point = RECORD
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Surface = RECORD
        Nor      : Vecteur;
        Cen      : Point;
        Taille   : INTEGER;
        NbSomLie : 1..20;
        SomLie   : ARRAY[1..20] OF 1..100;
        NbSurAdj : 1..20;
        SurAdj   : ARRAY[1..20] OF 1..20;
        END;
  Objet = RECORD
        NbSom : 1..100;
        Som   : ARRAY[1..100] OF Point;
        NbSur : 1..20;
        Sur   : ARRAY[1..20] OF Surface;
        END;
  PairSurMod = RECORD;
        AngMod   : Degre;
        DisMod   : INTEGER;
        ConvMod  : BOOLEAN;
        Im1,Im2  : 1..20;
        D1,D2    : REAL;
        Nm1,Nm2  : Vecteur;
        Sm1,Sm2  : INTEGER;
        END;


{====================== DECLARATION DE VARIABLES ==========================}

VAR
  Obj                 : Objet;
```

```
  SurMod               : PairSurMod;
  IndSur,IndSurAdj     : 1..20;
  IndSom1,IndSom2      : 1..100;
  Som1,Som2            : 1..100;
  NomficObj,NomFicMod  : PACKED ARRAY[1..12] OF CHAR;
  FicModele            : FILE OF PairSurMod;
  FicObjet             : FILE OF Objet;
  Choix                : CHAR;
  Pc                   : Point;
  Dis                  : INTEGER;
  Vect                 : Vecteur;


{======================= DECLARATION DE PROCEDURES ======================}

FUNCTION ProdScal(V1,V2 : Vecteur) : REAL;                          EXTERN;
FUNCTION ArcCos(Alpha : REAL) : Degre;                              EXTERN;
FUNCTION DistancePlan(Pc,Ps : Point; N : Vecteur) : REAL;           EXTERN;
FUNCTION CarreDistPoint(P1,P2 : Point) : INTEGER;                   EXTERN;


{*************************************************************************}
{*                          PROGRAMME PRINCIPAL                          *}
{*************************************************************************}

BEGIN
WRITELN('                    **** MODELISATION D''UN OBJET ****');
REPEAT
  WRITELN;WRITELN;WRITELN;
  WRITE(' Nom du fichier Objet ? ');READLN(NomFicObj);
  OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
  WRITELN;CLOSE(FicObjet);
  WRITE(' Nom du fichier Modele ? ');READLN(NomFicMod);
  OPEN(FicModele,NomFicMod);REWRITE(FicModele);
  WITH Obj DO
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur],SurMod DO
        FOR IndSurAdj := 1 TO NbSurAdj  DO
          IF IndSur<SurAdj[IndSurAdj] THEN
            BEGIN DisMod:=0;
            AngMod := ArcCos(ProdScal(Nor,Sur[SurAdj[IndSurAdj]].Nor));
            WITH Vect,Cen DO BEGIN u:=x; v:=y; w:=z; END;
            D1:=DistancePlan(Pc,Cen,Nor); Nml:=Nor; Iml:=IndSur; Sml:=Taille;
            WITH Sur[SurAdj[IndSurAdj]] DO
              BEGIN D2:=DistancePlan(Pc,Cen,Nor);
```

```
                Nm2:=Nor; Im2:=Sur[IndSur].SurAdj[IndSurAdj]; Sm2:=Taille;
                WITH Vect,Cen DO BEGIN u:=u-x; v:=v-y; w:=w-z; END;
                IF ProdScal(Nor,Vect)>0 THEN ConvMod:=FALSE ELSE ConvMod:=TRUE;
                END;
              FOR IndSom1:=1 TO Sur[IndSur].NbSomLie DO
                FOR IndSom2:=1 TO Sur[SurAdj[IndSurAdj]].NbSomLie DO
                  BEGIN Som1:=Sur[IndSur].Somlie[IndSom1];
                  Som2:=Sur[SurAdj[IndSurAdj]].SomLie[IndSom2];
                  Dis:=CarreDistPoint(Som[Som1],Som[Som2]);
                  If Dis>DisMod THEN DisMod:=Dis;
                  END;
              WRITE(FicModele,SurMod);
              END;
  CLOSE(FicModele);
  WRITELN;WRITELN;WRITE(' Autre objet a modeliser ? ');READLN(Choix);
UNTIL (Choix='n') OR (Choix='N');
END.
```

E. "CREATIONIMAGE" PROGRAM

This program allows to create an artificial scene containing many polyhedra. It uses the CAD-Models of the polyhedra and creates all the data necessary for the complete analysis of this scene.

Parameters

- Name of the artificial image. This name is not followed by any extension (e.g., ARTIIMAGE).

- Millimeter distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis) and between two levels of the elevation image axis (Z-axis).

- Names of the CAD-Models of the polyhedra.

- Values of the translations to be applied to the CAD-Models.

- Values of the rotations to be applied to the CAD-Models.

Input Files

- Files containing the CAD-Models of the polyhedra.

Output Files

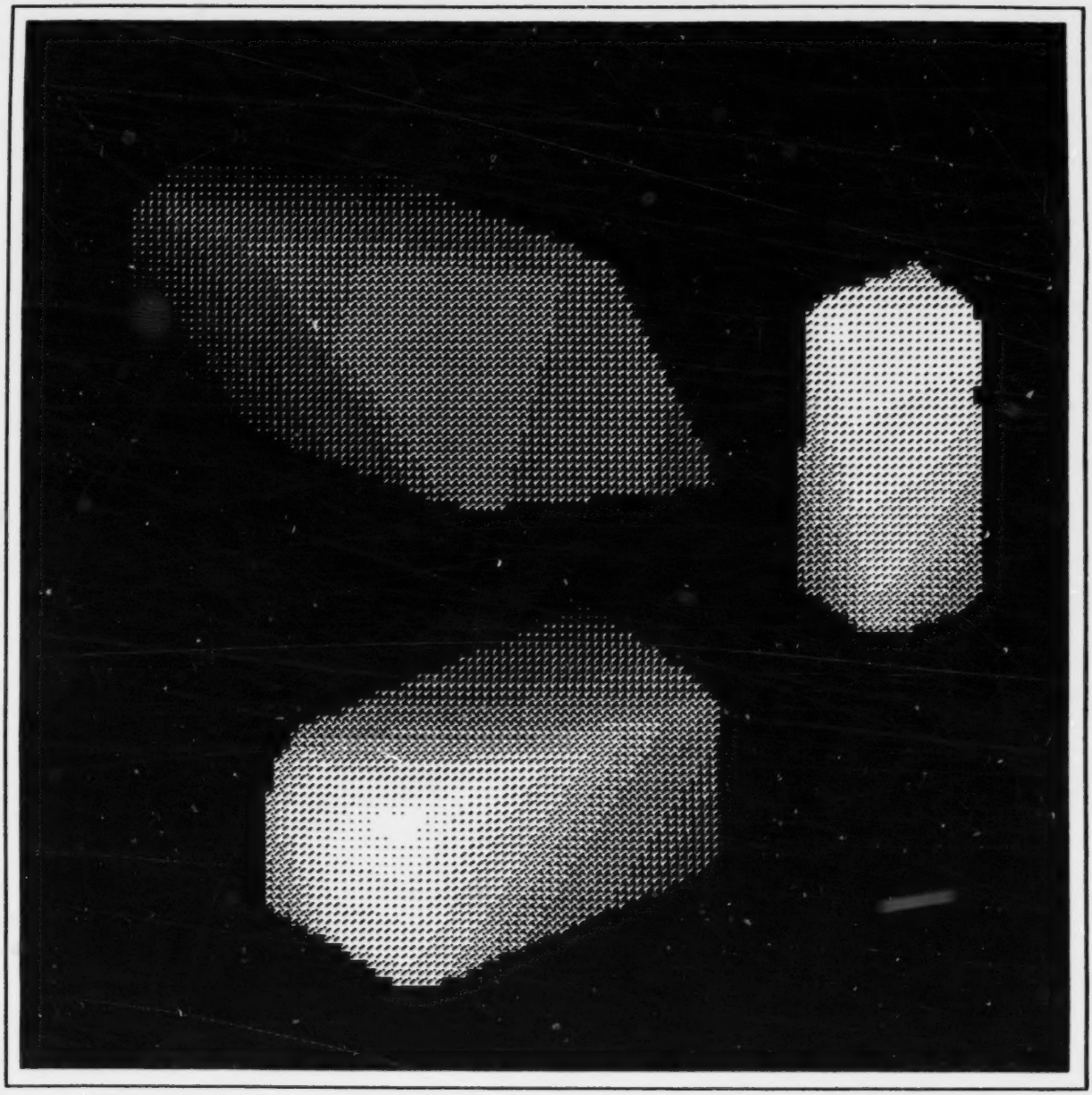
- Range data image of the artificial scene (see Fig. A-3a).

- Location of the different polyhedron (see Fig. A-3b). This location corresponds to the image obtained by thresholding the range data.

- Location of the different plane surfaces of the polyhedra. These data corresponds to an image where each point belonging to one surface of a polyhedron is labelled by the index of this surface (see Fig. A-3c).

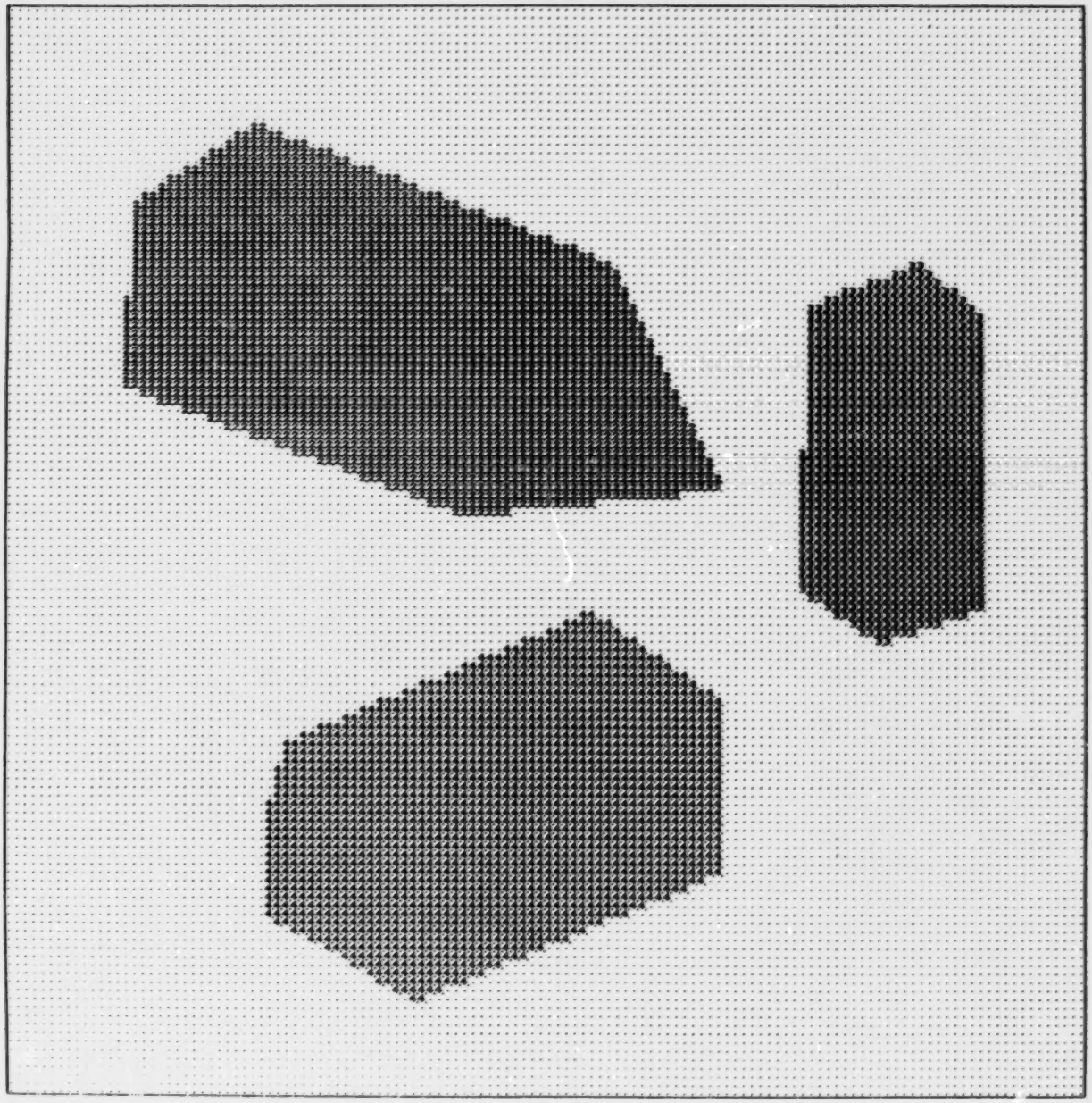
- Adjacency of the different plane surfaces. To create this image, the edges between the plane surfaces are labelled by a special value (see Fig. A-3d).

note : These files are automatically created by the program. For this reason, the name of these files is equal to the name of the artificial image followed respectively by the extensions ".DIS", ".SEU", ".SPL" and ".SCO" (e.g., ARTIIMAGE.DIS).



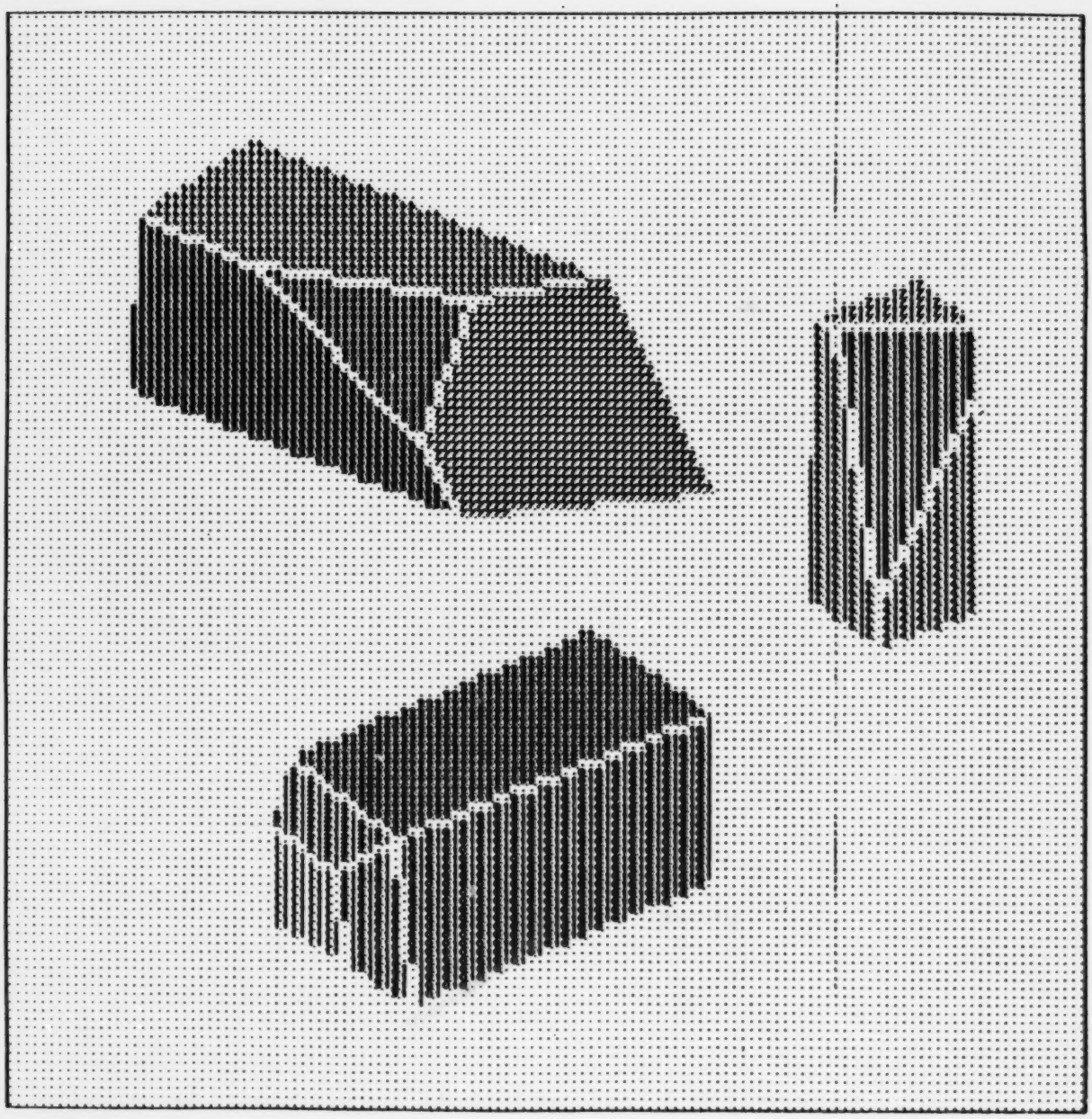
(a)

Figure A-3. Output files of the "CREATIONIMAGE" program. (a) Range data of an artificial image. (b) Location of the different polyhedra. (c) Location of the different plane surfaces. (d) Adjacency of the different plane surfaces.



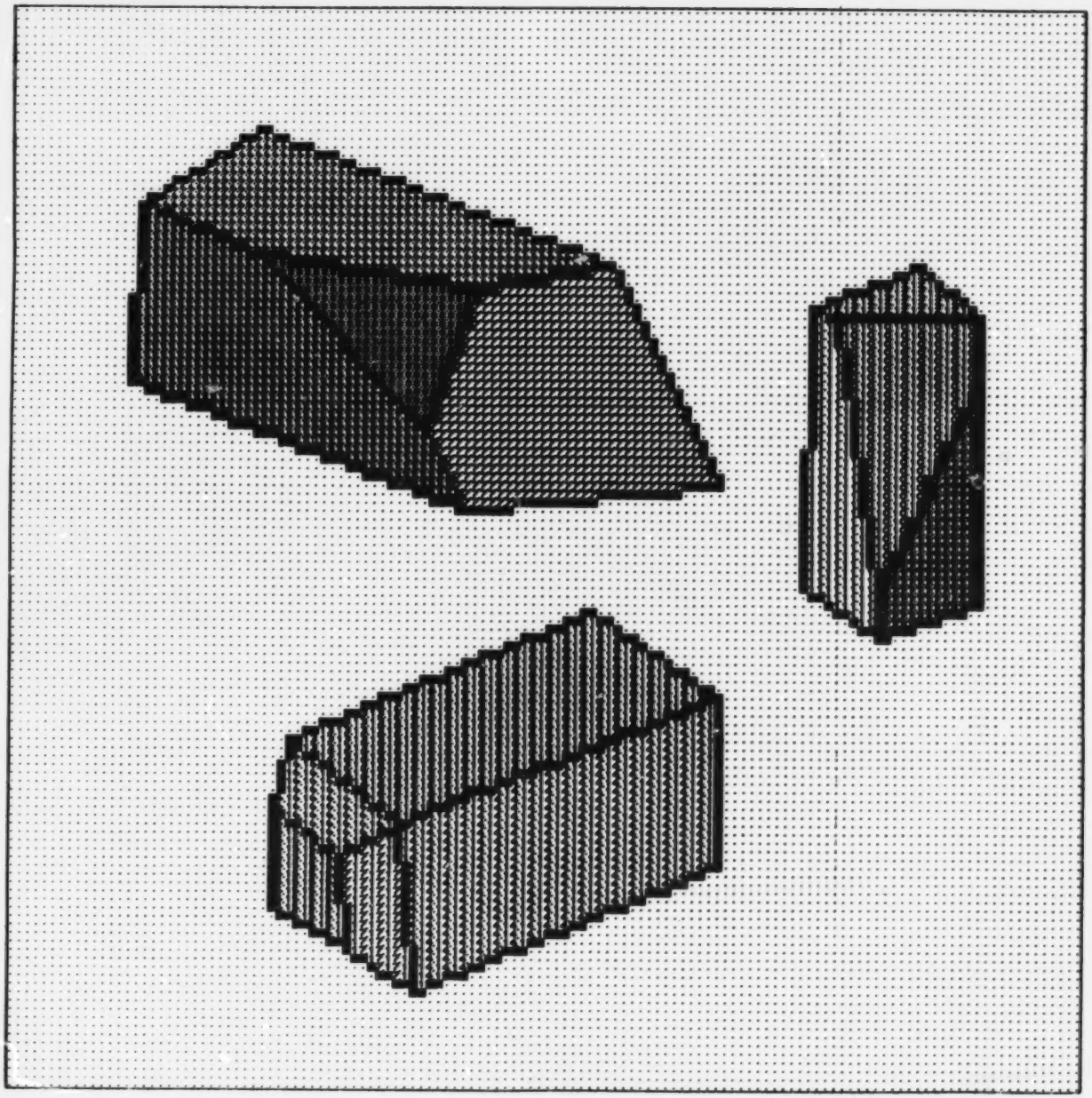
(b)

Figure A-3. (Continued).



(c)

Figure A-3. (Continued).



(d)

Figure A-3. (Concluded).

```
{*****************************************************************************}
                        PROGRAM CreationImage(INPUT,OUTPUT);
{*****************************************************************************}


{======================== DECLARATION DE TYPES ========================}

TYPE
  Bytes       = -32766..32765;
  Ligne       = PACKED ARRAY[0..127] OF Bytes;
  Plan        =  ARRAY[0..127] OF ligne;
  FichierPlan = FILE OF Ligne;
  Degre       = -180..180;
  Point = RECORD
        x,y,z    : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w    : REAL;
        END;
  Rotation = RECORD
        a,b,g    : INTEGER;
        END;
  Surface = RECORD
        Nor      : Vecteur;
        Cen      : Point;
        Taille   : INTEGER;
        NbSomLie : 1..20;
        SomLie   : ARRAY[1..20] OF 1..100;
        NbSurAdj : 1..20;
        SurAdj   : ARRAY[1..20] OF 1..20;
        END;
  Objet = RECORD
        NbSom    : 1..100;
        Som      : ARRAY[1..100] OF Point;
        NbSur    : 1..20;
        Sur      : ARRAY[1..20] OF Surface;
        END;
  Nom = VARYING[12] OF CHAR;


{====================== DECLARATION DE VARIABLES ======================}

VAR
  FicDis,FicBloc,FicSurfP,FicSurfC              : FichierPlan;
  NomFicObj,NomFicDis,NomIma,NomFicBloc         : Nom;
```

```
  NomFPSurf,NomFCSurf                          : Nom;
  Pt0,Pt1,Pt2,Mil                              : Point;
  N0,Nt1,Nt2                                   : Vecteur;
  Obj                                          : Objet;
  IndSom,IndSomLie                             : 1..100;
  FicObjet                                     : FILE OF Objet;
  Rot                                          : Rotation;
  Tra                                          : Vecteur;
  TabDist,TabBloc,TabSurfP,TabSurfC            : Plan;
  CoefX,CoefY,CoefZ                            : REAL;
  Choix                                        : CHAR;
  IndObj,NumSur                                : INTEGER;

{====================== DECLARATION DE PROCEDURES ======================}

PROCEDURE NormerVec(VAR V : Vecteur);                                EXTERN;
PROCEDURE RotationVect(Rot : Rotation; VAR Vec : Vecteur);           EXTERN;
PROCEDURE RotationPoint(Rot : Rotation; VAR P : Point);              EXTERN;
FUNCTION  ProdScal(V1,V2 : Vecteur):REAL;                            EXTERN;
FUNCTION  ArcCos(Alpha : REAL) : Degre;                              EXTERN;
PROCEDURE Translation(Tra : Vecteur; Var P : Point);                 EXTERN;
PROCEDURE ProdVect(V1,V2 : Vecteur; VAR V0 : Vecteur);               EXTERN;
PROCEDURE ProdVectNor(V1,V2 : Vecteur; VAR V0 : Vecteur);            EXTERN;
PROCEDURE EcritureImage(VAR Fic : FichierPlan; Tab : Plan);          EXTERN;

{----------------------------------------------------------------------}
                        PROCEDURE Initialisation;
{----------------------------------------------------------------------}

VAR I,J : INTEGER;

BEGIN
WRITELN('                        **** Programme Creation d''Image ****');
WRITELN;WRITELN;
WRITE(' Nom de l''image cree ? ');READLN(NomIma);
NomFicDis:=NomIma+'.DIS';OPEN(FicDis,NomFicDis,NEW);REWRITE(FicDis);
NomFicBloc:=NomIma+'.SEU';OPEN(FicBloc,NomFicBloc,NEW);REWRITE(FicBloc);
NomFPSurf:=NomIma+'.SPL';OPEN(FicSurfP,NomFPSurf,NEW);REWRITE(FicSurfP);
NomFCSurf:=NomIma+'.SCO';OPEN(FicSurfC,NomFCSurf,NEW);REWRITE(FicSurfC);
FOR J:=0 TO 127 DO TabDist[0,J]:=0;
FOR I:=1 TO 127 DO TabDist[I]:=TabDist[0];
TabBloc:=TabDist; TabSurfP:=TabDist; TabSurfC:=TabDist;
WRITE(' CoefX,CoefY,CoefZ ? ');READLN(CoefX,CoefY,CoefZ);
IndObj:=1; NumSur:=2;
END;
```

```
{-------------------------------------------------------------------------}
                    PROCEDURE CalculProjectionObjet;
{-------------------------------------------------------------------------}

TYPE
  Pixel = RECORD
        x,y : INTEGER;
        Suivant : Pixel;
        END;

VAR
  IndSur : 1..20;
  IndSom,IndSomLie : 1..100;
  P1,P2,PBloc : Point;
  TabXY :Plan;
  Hauteur : INTEGER;
  Pi0,Pi1,Pi2,Debut : Pixel;
  Ind,Xp,Yp,NumBloc : INTEGER;
  Vec,Vert,V1,V2 : Vecteur;
  I,J : INTEGER;

PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER; VAR Tab:PLAN);EXTERN;

BEGIN
WITH Obj DO
  BEGIN
  FOR IndSom:=1 TO NbSom DO
    WITH Som[IndSom] DO
      BEGIN RotationPoint(Rot,Som[IndSom]); Translation(Tra,Som[IndSom]);
      x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);
      END;
  FOR IndSur :=1 TO NbSur DO
    WITH Sur[IndSur] DO
      BEGIN RotationVect(Rot,Nor);
      WITH Nor DO
        BEGIN u:=u/(CoefY*CoefZ); v:=v/(CoefX*CoefZ); w:=w/(CoefX*CoefY);
        END;
      NormerVec(Nor);
      IF Nor.w > 0.1 THEN
        BEGIN NumSur:=NumSur+1;
        FOR IndSomLie:= 1 TO NbSomLie DO
          BEGIN P1:=Som[Somlie[IndSomlie]];
          IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
          ELSE P2:=Som[Somlie[IndSomlie+1]];
          ProjecSegment(P1,P2,1,TabSurfC);
```

```
          ProjecSegment(P1,P2,IndObj+1,TabBloc);
          END;
        RotationPoint(Rot,Cen); Translation(Tra,Cen);
        WITH Cen DO
          BEGIN x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);END;
        IF(Cen.x>=0)AND(Cen.x<=127)AND(Cen.Y>=0)AND(Cen.Y<=127)THEN
        IF(TabSurfC[Cen.x,Cen.y]=0) THEN
          BEGIN NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
          WITH Cen DO
            BEGIN Pi0.x:=x; Pi0.y:=Y; TabDist[x,y]:=z*100;
            TabBloc[x,y]:=IndObj+1; TabSurfP[x,y]:=NumSur+2;
            TabSurfC[x,y]:=NumSur+2;
            END;
          WHILE Pi0.Suivant<>NIL DO
            WITH Pi0 DO
              BEGIN
              FOR Ind:=1 TO 4 DO
                BEGIN
                CASE Ind OF
                  1: BEGIN Xp:=x+1; Yp:=y; END;
                  2: BEGIN Xp:=x-1; Yp:=y; END;
                  3: BEGIN Xp:=x; Yp:=y+1; END;
                  4: BEGIN Xp:=x; Yp:=y-1; END;
                  END;
                IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
                  IF TabSurfC[Xp,Yp]<=1  THEN
                    BEGIN
                    IF TabSurfC[Xp,Yp]=0 THEN
                      BEGIN Pi1.x:=Xp;Pi1.y:=Yp;NEW(Pi2);Pi1.Suivant:=Pi2;
                      Pi2.Suivant:=NIL; Pi1:=Pi2; TabSurfC[Xp,Yp]:=NumSur+2;
                      TabBloc[Xp,Yp]:=IndObj+1; TabSurfP[Xp,Yp]:=NumSur+2;
                      END;
                    WITH Vec,Cen DO BEGIN u:=(x-Xp); v:=(y-Yp); w:=z; END;
                    TabDist[Xp,Yp]:=ROUND((ProdScal(Vec,Nor)*100)/Nor.w);
                    END;
                  END;
                Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
                END;
            DISPOSE(Pi0);
            END;
          END;
        END;
      END;
    END;
  END;
END;
```

```
{=============================================================================}
{=============================  PROCEDURE SauveImage;  =======================}
{=============================================================================}
VAR
  I,J : INTEGER;

BEGIN
EcritureImage(FicBloc,TabBloc); EcritureImage(FicDis,TabDist);
EcritureImage(FicSurfP,TabSurfP);
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO
    IF TabSurfC[I,J]=0 THEN TabSurfC[I,J]:=2;
EcritureImage(FicSurfC,TabSurfC);
END;


{*****************************************************************************}
{*****************************  PROGRAMME PRINCIPAL  *************************}
{*****************************************************************************}
BEGIN
Initialisation;
REPEAT
  WRITELN;WRITELN;
  WRITE(' Nom du fichier objet ? '); READLN(NomFicObj);
  OPEN(FicObjet,NomFicObj,OLD); RESET(FicObjet);READ(FicObjet,Obj);
  WRITE(' Translation ? '); WITH Tra DO READLN(u,v,w);
  WRITE(' Rotation ? '); WITH Rot DO READLN(a,b,g);
  CalculProjectionObjet;
  CLOSE(FicObjet); IndObj:=IndObj+1;
  WRITE(' autre objet a projeter ? ');READLN(Choix);
UNTIL (Choix='n') OR (Choix='N');
SauveImage;
END.
```

F. "LECTURESCENE" PROGRAM

This program is used to extract the characteristics of the studied scene. Some input data come from the pre-processing step. For more informations about these data see companion paper (13).

Parameters

- Name of the studied image. This name is not followed by any extension (e.g., IMAGE02).

- Millimetre distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis) and between two levels of the range data (Z-axis).

- Minimum size for a plane surface (number of pixels).

- Minimum difference of height between two neighbouring pixels will correspond to the presence of a vertical surface (number of millimetres).

Input Files

- Range data image of the studied scene (see Fig. A-4a).

- Location of the different groups of overlapping objects (see Fig. A-4b). This location corresponds to the image obtained by thresholding the range data.

- Location of the different plane surfaces extracted from the range data. These data correspond to an image where each point belonging to one surface is labelled by the index of this surface (see Fig. A-4c).

- Adjacency of the different plane surfaces. To create this image, the plane surfaces of the previous image have been expanded and the edges in between are labelled with a special value (see Fig. A-4d).

note: These files are automatically read by the program. For this reason, the name of these files must be equal to the name of the studied image followed respectively by the extensions ".DIS", ".SEU", ".SPL" and ".SCO" (e.g., IMAGE02.DIS).

Output Files

- File containing the characteristics of each pair of adjacent plane

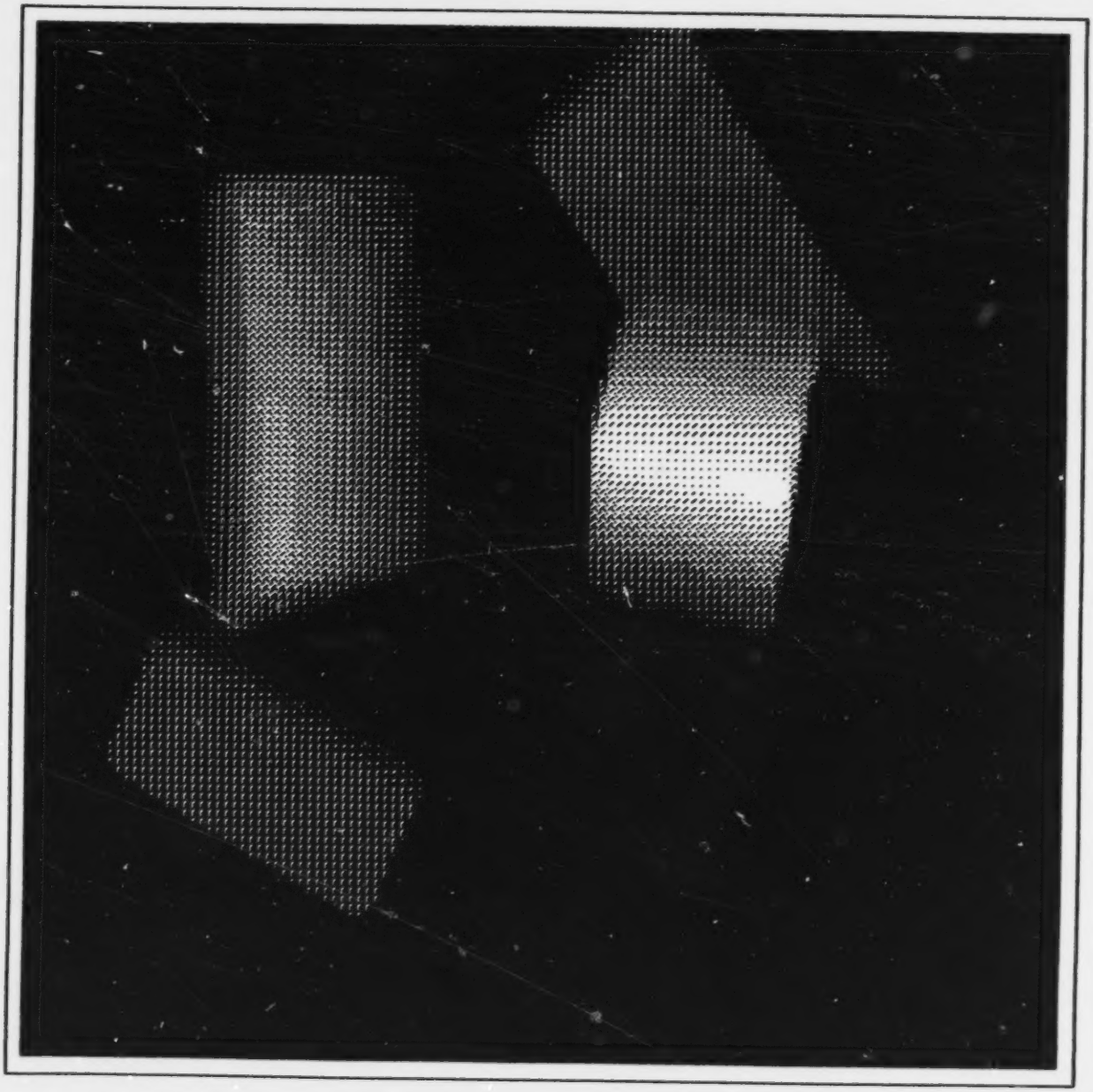
surfaces of the studied scene (for more details see paragraph II.2).

- New labelling of the plane surfaces after cancellation of the too small ones (see Fig. A-5a).

- Vertical surfaces. These data correspond to an image where each pixel belonging to a vertical surface is labelled with the index of this vertical surface (see Fig. A-5b).

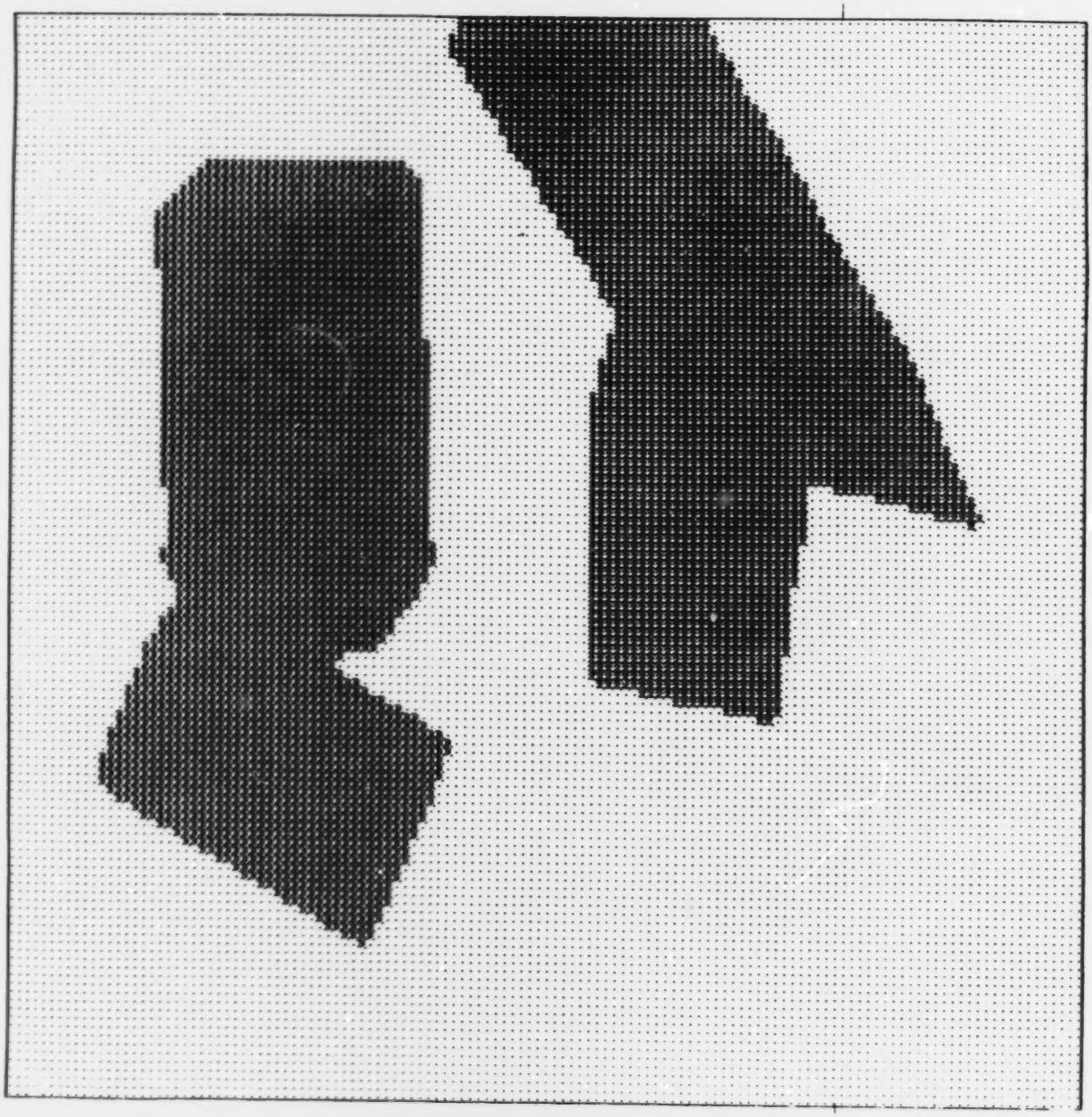
- Modified range data (see Fig. A-5c). The distance between the background and the objects is increased to penalise the bad matches between models and scene objects (see paragraph II.7).

note: These files are automatically created by the program and their names are equal to the name of the image followed respectively by the extensions ".CSA", ".SET", ".SVR" and ".DIM" (e.g., IMAGE02.CSA).



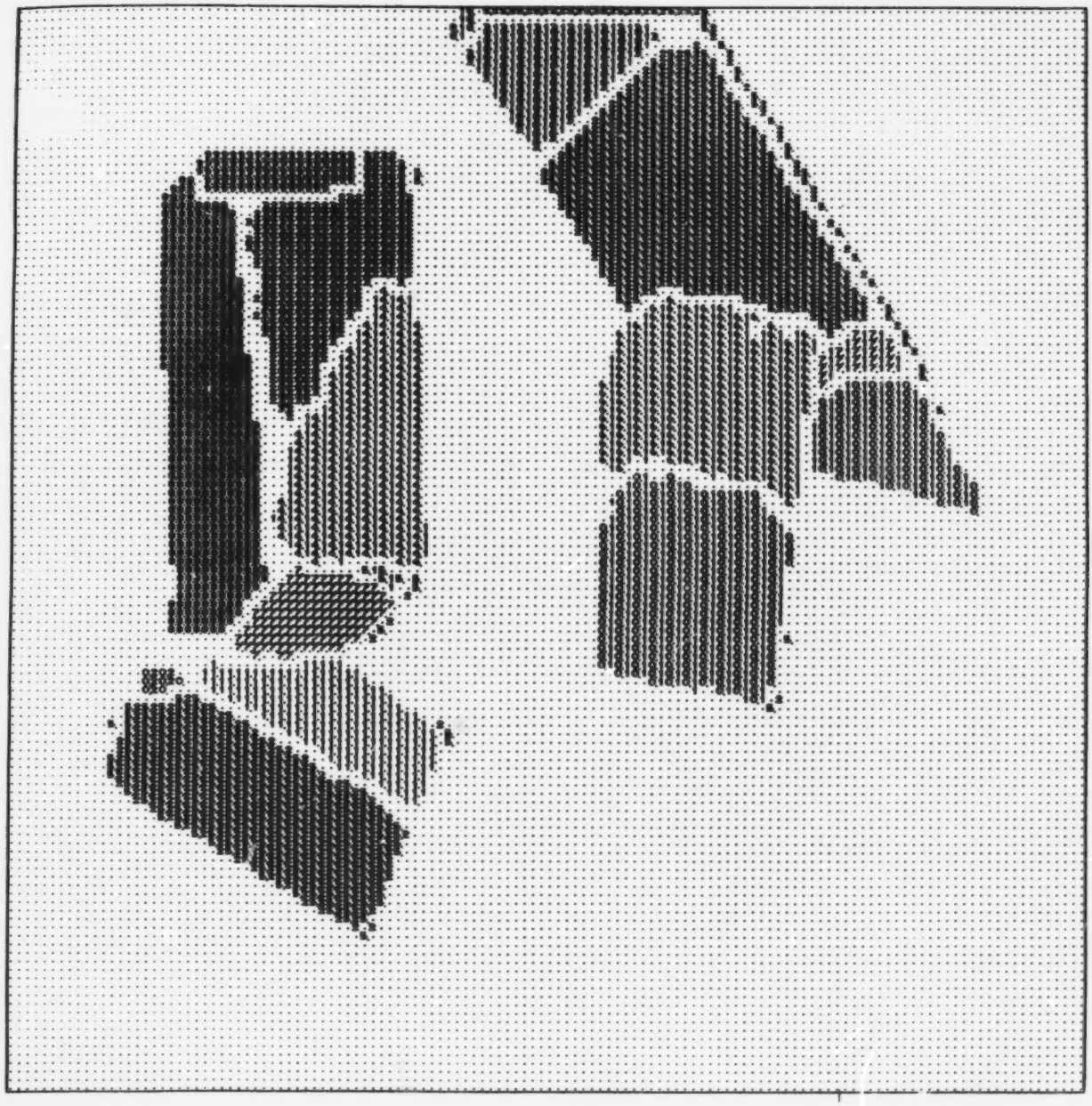
(a)

Figure A-4. Input files of the "LECTURESCENE" program. (a) Range data. (b) Location of the different groups of overlapping objects. (c) Location of the different plane surfaces. (d) Adjacency of the different plane surfaces.



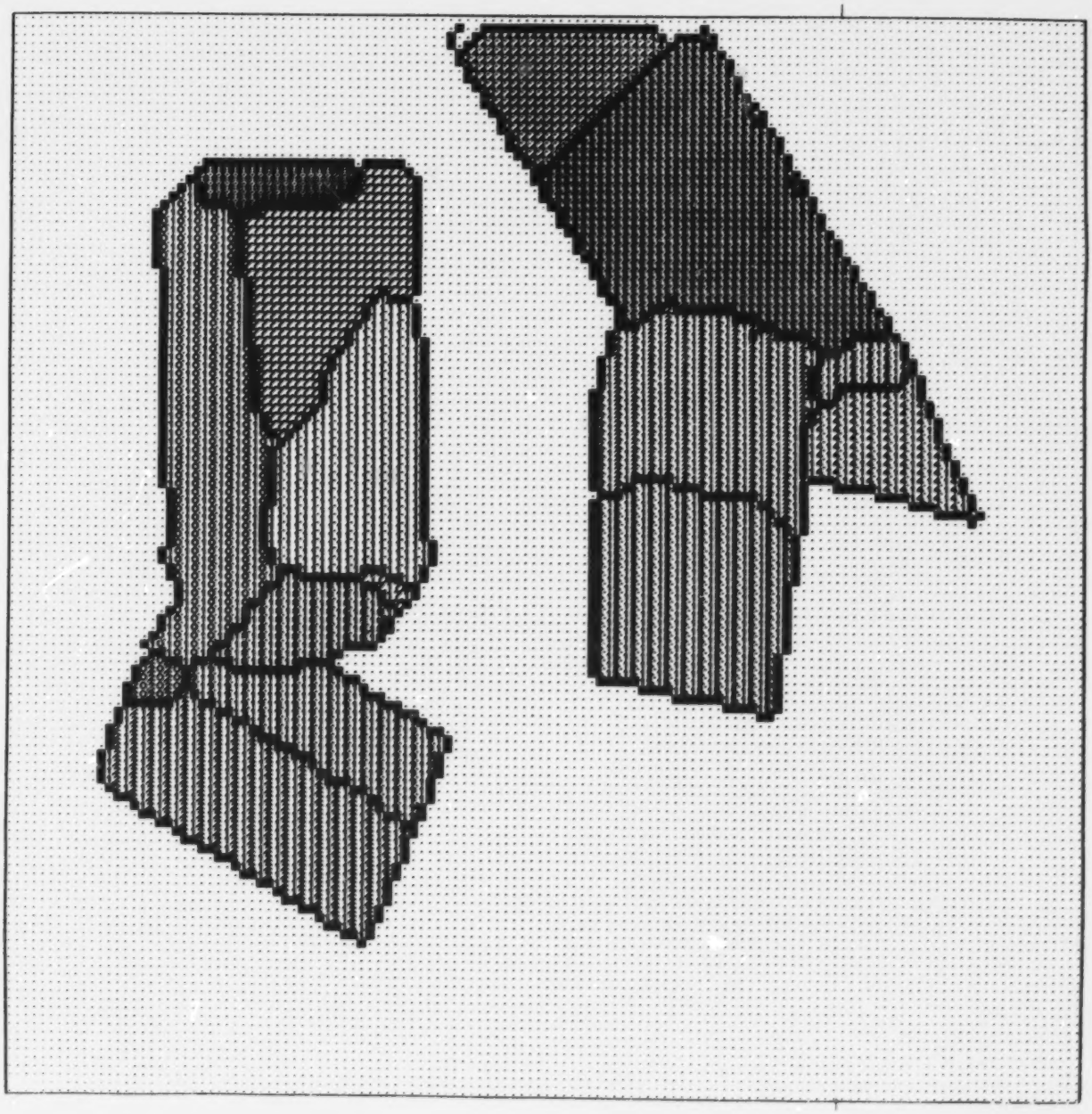
(b)

Figure A-4. (Continued).



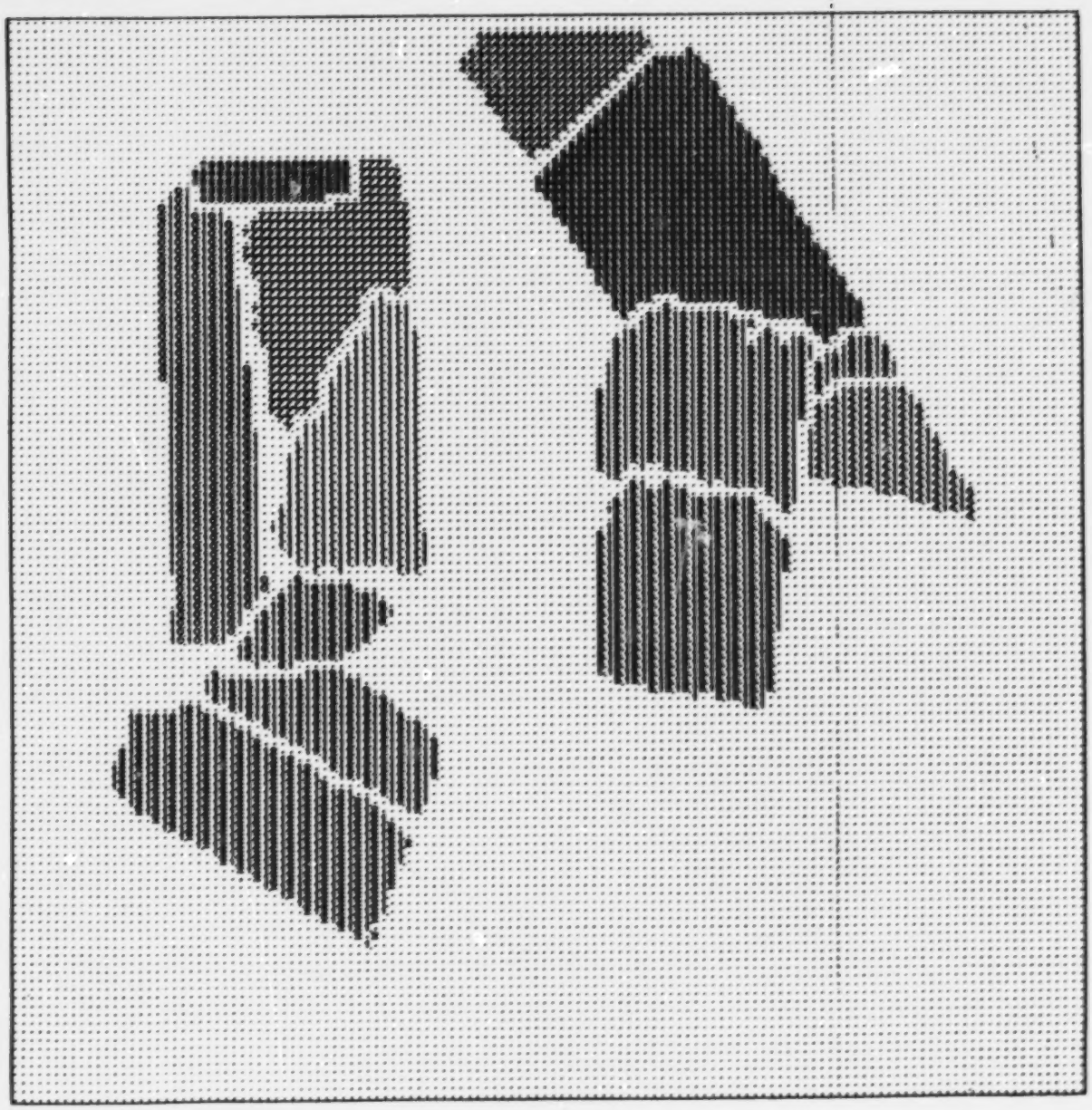
(c)

Figure A-4. (Continued).



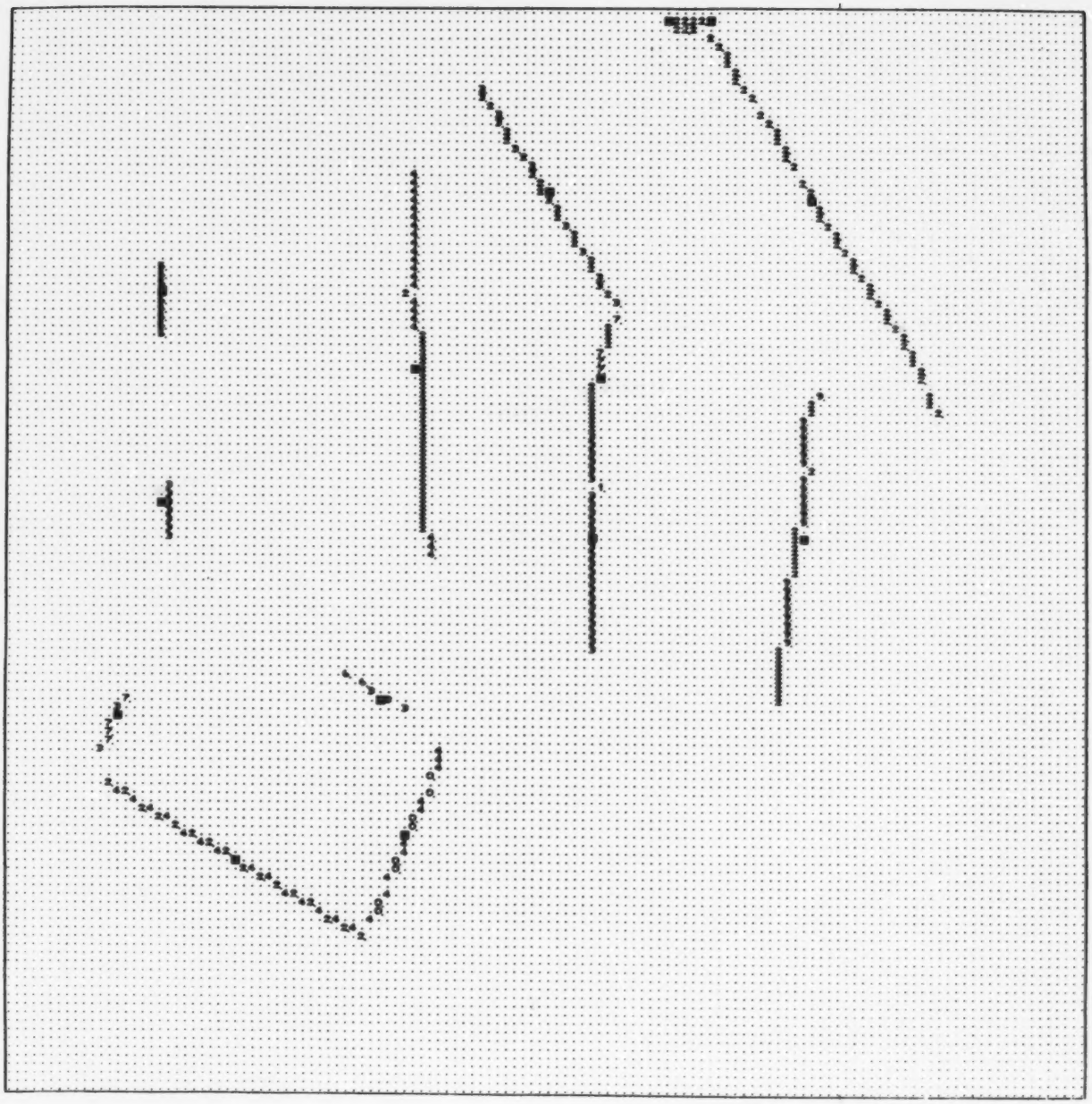
(d)

Figure A-4. (Concluded).



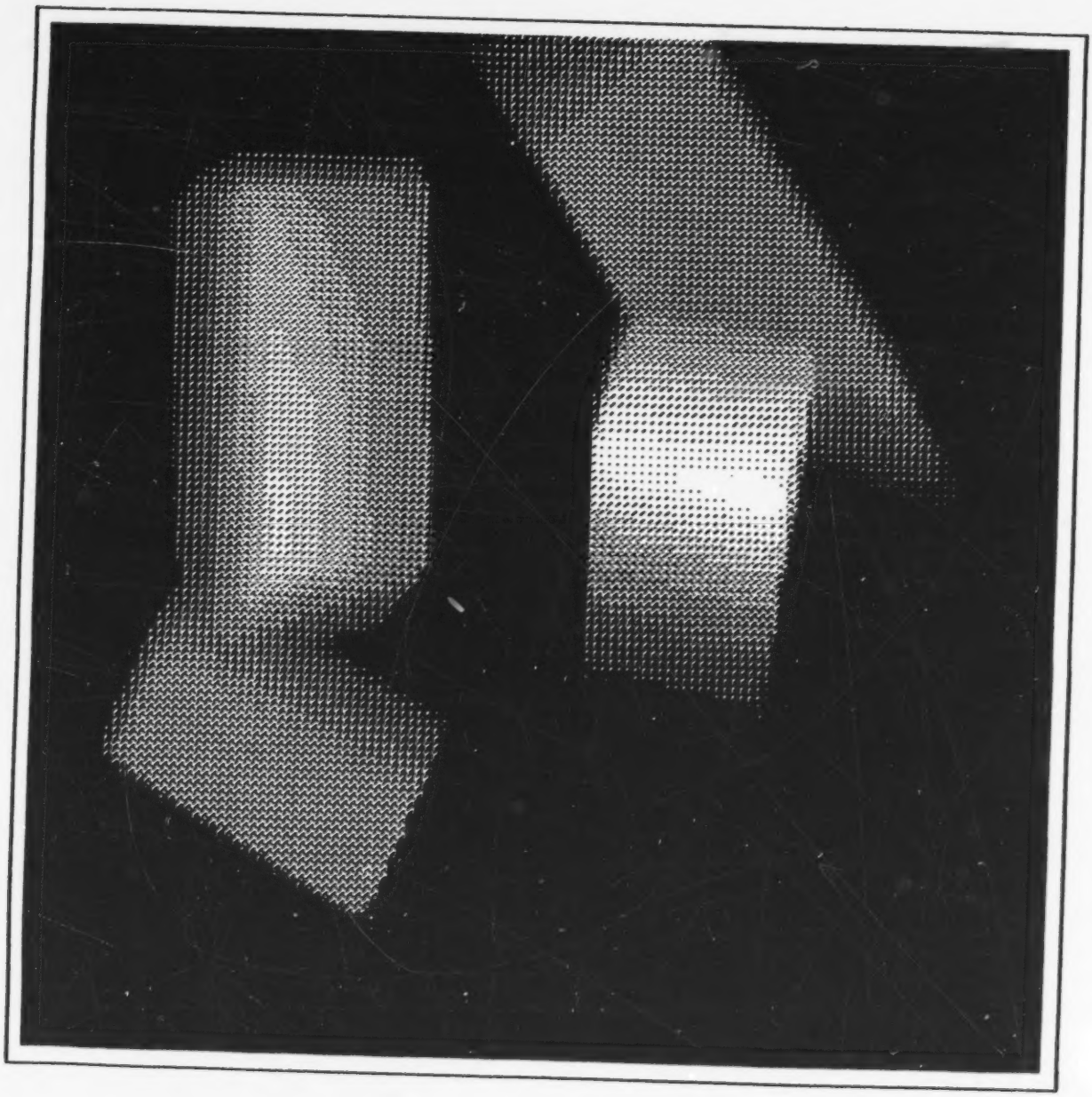
(a)

Figure A-5. Output files of the "LECTURESCENE" program. (a) New labelling of the plane surfaces. (b) Location of the vertical surfaces. (c) Modified range data.



(b)

Figure A-5. (Continued).



(c)

Figure A-5. (Concluded).

```
{*****************************************************************************}
                    PROGRAM LectureScene(INPUT,OUTPUT);
{*****************************************************************************}


{====================== DECLARATION DE CONSTANTES ======================}

CONST
  CoefPi180 = 0.01745329238;
  Coef180Pi = 57.29577637;

{======================== DECLARATION DE TYPES ========================}

TYPE
  Bytes       = -32766..32765;
  Ligne       = PACKED ARRAY[0..127] OF Bytes;
  Plan        = ARRAY[0..127] OF Ligne;
  FichierPlan = FILE Of Ligne;
  Degre       = -180..180;
  Point = RECORD;
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Surface = RECORD
        Cen,Ori     : Point;
        Nor         : Vecteur;
        NumBloc     : INTEGER;
        HGrad,Vgrad : INTEGER;
        NbPt,Taille : INTEGER;
        END;
  PairSurIm = RECORD;
        AngIm   : Degre;
        DisIm   : INTEGER;
        InBloc  : INTEGER;
        ConvIm  : BOOLEAN;
        I11,I12 : INTEGER;
        P1,P2   : Point;
        N11,N12 : Vecteur;
        S11,S12 : INTEGER;
        END;
  NomFic = VARYING[12] OF CHAR;
  Pixel = RECORD
        x,y : INTEGER;
```

```
        Suivant : Pixel;
        END;

{========================= DECLARATION DE VARIABLES =========================}

VAR
  FicSeuil,FicSurfC,FicSurfP,FicSurEt,FicDist : FichierPlan;
  FicSurfV                                    : FichierPlan;
  FicSurIm                                    : FILE OF PairSurIm;
  NomFicSeuil,NomFCSurf,NomFPSurf,NomIma      : NomFic;
  NomFESurf,NomFicDist,NomFicSurIm,NomFVSurf  : NomFic;
  TabSeuil,TabSurfC,TabSurfP,TabDist          : Plan;
  TabDrDis,TabDrOri                           : Plan;
  I,J,K,L,Ind1,Ind2                           : INTEGER;
  Sur                                         : ARRAY[1..200] OF Surface;
  Adj                                         : ARRAY[1..4] OF INTEGER;
  IndSurf,IndSurfP,NbSurf,NbSurfNV            : INTEGER;
  IndSurfAdj,NbSurfAdj,InSA1,InSA2            : INTEGER;
  NouvelleSurf,SurfConnec                     : BOOLEAN;
  SurIm                                       : PairSurIm;
  Alpha,Beta                                  : REAL;
  Pi0,Pi1,Pi2,Debut                           : Pixel;
  NbPtC,Ind,Xp,Yp                             : INTEGER;
  IndSol,NbBloc,IndBloc                       : INTEGER;
  NbCSA                                       : INTEGER;
  SeuilSurf,SeuilSaut                         : INTEGER;
  TrouvePt                                    : BOOLEAN;
  CoefX,CoefY,CoefZ                           : REAL;
  Accu,Classe : PACKED ARRAY[-64..63,-64..63] OF -128..127;
  TAdj        : PACKED ARRAY[1..200,1..200] OF BOOLEAN;
  Ht          : ARRAY[-1..1,-1..1] OF INTEGER;


{======================= DECLARATION DE PROCEDURES =========================}

PROCEDURE LectureImage( VAR Fic : FichierPlan; Tab : Plan);        EXTERN;
PROCEDURE EcritureImage( VAR Fic : FichierPlan; Tab : Plan);       EXTERN;
FUNCTION  ArcCos(Alpha : REAL) : Degre;                            EXTERN;
FUNCTION  ProdScal(V1,V2 : Vecteur) : REAL;                        EXTERN;
FUNCTION  CarreDistPoint(P1,P2 : Point) : INTEGER;                 EXTERN;
PROCEDURE NormerVec(VAR V:Vecteur);                                EXTERN;
```

```
{=====================================================================}
      FUNCTION  LissageMed(x,y : INTEGER; VAR Tab : Plan):INTEGER;
{=====================================================================}

VAR
  Valeur,NbCaseExa : INTEGER;
  V                : ARRAY[1..10] OF INTEGER;
  Decale           : BOOLEAN;
  Ind1,Ind2,I,J    : INTEGER;

BEGIN V[1]:=-1;
FOR I:=-1 TO 1 DO
  FOR J:=-1 TO 1 DO
    BEGIN Valeur:=Tab[x+I,y+J]; NbCaseExa:=(1+I)*3+2+J;
    Ind1:=1;Decale:=FALSE;
    REPEAT
      IF V[Ind1]<Valeur THEN
        BEGIN FOR Ind2:= NbCaseExa+1 DOWNTO Ind1+1 DO V[Ind2]:=V[Ind2-1];
        V[Ind1]:=Valeur;Decale:=TRUE;
        END;
      Ind1:=Ind1+1;
    UNTIL (Ind1>5)OR(Decale);
    END;
LissageMed:=V[5];
END;


{=====================================================================}
                        PROCEDURE Initialisation;
{=====================================================================}

BEGIN
WRITELN(' **** Programme de lecture d'+'une scene **** ');
WRITELN;WRITELN;WRITELN;
WRITE(' Nom de l'+'image a traiter ? ');READLN(NomIma);WRITELN;
NomFicSeuil:=NomIma+'.SEU';
OPEN(FicSeuil,NomFicSeuil,OLD);LectureImage(FicSeuil,TabSeuil);
NomFPSurf:=NomIma+'.SPL';
OPEN(FicSurfP,NomFPSurf,OLD);LectureImage(FicSurfP,TabSurfP);
NomFCSurf:=NomIma+'.SCO';
OPEN(FicSurfC,NomFCSurf,OLD);LectureImage(FicSurfC,TabSurfC);
NomFicDist:=NomIma+'.DIS';
OPEN(FicDist,NomFicDist,OLD);LectureImage(FicDist,TabDist);
NomFicSurIm:=NomIma+'.CSA';
OPEN(FicSurIm,NomFicSurIm);REWRITE(FicSurIm);
```

```
NomFESurf:=NomIma+'.SET'; OPEN(FicSurEt,NomFESurf,NEW);
NomFVSurf:=NomIma+'.SVR'; OPEN(FicSurfV,NomFVSurf,NEW);
CLOSE(FicDist); NomFicDist:=NomIma+'.DIM'; OPEN(FicDist,NomFicDist,NEW);
WRITE('CoefX,CoefY,CoefZ ?');READLN(CoefX,CoefY,CoefZ);WRITELN;
WRITE('Taille mini surfaces planes (nb pixels) ?');
READLN(SeuilSurf);WRITELN;
WRITE('Saut mini pour creer surfaces verticales (mm) ?');
READLN(SeuilSaut);WRITELN;
FOR InSAl:=1 TO 200 DO TAdj[1,InSAl]:=FALSE;
FOR InSAl:=2 TO 200 DO TAdj[InSAl]:=TAdj[1];
END;


{=============================================================================}
                          PROCEDURE RecherIndiceSol;
{=============================================================================}

VAR
  Haut,HautMini : INTEGER;

BEGIN
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabSeuil[I,J]=0 THEN
      BEGIN Haut:=LissageMed(I,J,TabDist);
      IF Haut<HautMini THEN HautMini:=Haut;
      END;
HautMini:=HautMini-10;
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO
    IF TabDist[I,J]<=HautMini THEN TabDist[I,J]:=10
    ELSE TabDist[I,J]:=TabDist[I,J]-HautMini;
IndSol:=0;I:=0;
REPEAT I:=I+1;J:=0;
  REPEAT J:=J+1;
    IF TabSeuil[I,J]=0 THEN
      BEGIN IndSol:=TabSurfC[I,J];
      WITH Sur[IndSol],Cen,Nor DO
        BEGIN x:=ROUND(I*CoefX); y:=ROUND(J*CoefY); z:=TabDist[I,J] DIV 100;
        z:=z+100; u:=0; v:=0; w:=1; NbPt:=99999;
        END;
      END;
  UNTIL (IndSol<>0) OR (J=127);
UNTIL (IndSol<>0) OR (I=127);
END;
```

```
{=====================================================================}
                    PROCEDURE CaractPlanesSurfaces;
{=====================================================================}

VAR
  Mx,Mxx,My,Myy,Mz,Mzz,Mxy,Mzx,Mzy : REAL;
  Vxx,Vyy,Vzz,Vxy,Vzx,Vzy           : REAL;
  Ht                                : INTEGER;

BEGIN NbSurf:=0;
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    BEGIN IndSurf:=TabSurfC[I,J]; IF IndSurf>NbSurf THEN NbSurf:=IndSurf;
    IF (IndSurf<>1)AND(IndSurf<>IndSol)AND(TabSurfP[I,J]>0) THEN
      BEGIN NEW(Pi0); NEW(Pi1); Pi0.x:=I; Pi0.y:=J; Pi0.Suivant:=Pi1;
      Pi1.Suivant:=NIL; IndSurfP:=TabSurfP[I,J]; TabSurfP[I,J]:=-IndSurf;
      Mx:=0; Mxx:=0; My:=0; Myy:=0; Mz:=0; Mzz:=0; Mxy:=0; Mzx:=0; Mzy:=0;
      WITH Sur[IndSurf],Ori DO
        BEGIN NumBloc:=TabSeuil[I,J]; x:=I; y:=J; Taille:=1; END;
      IF Sur[IndSurf].NumBloc>NbBloc THEN NbBloc:=Sur[IndSurf].NumBloc;
      WHILE Pi0.Suivant<>NIL DO
        WITH Pi0,Sur[IndSurf] DO
          BEGIN
          IF (TabSurfP[x-1,y-1]<>0)AND(TabSurfP[x,y-1]<>0)
          AND(TabSurfP[x+1,y-1]<>0)AND(TabSurfP[x+1,y]<>0)
          AND(TabSurfP[x+1,y+1]<>0)AND(TabSurfP[x,y+1]<>0)
          AND(TabSurfP[x-1,y+1]<>0)AND(TabSurfP[x-1,y]<>0) THEN
            BEGIN NbPt:=NbPt+1; Ht:=LissageMed(Pi0.x,Pi0.y,TabDist);
            Mx:=Mx+x; Mxx:=Mxx+x*x; My:=My+y; Myy:=Myy+y*y; Mz:=Mz+Ht;
            Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+x*y; Mzx:=Mzx+Ht*x; Mzy:=Mzy+Ht*y;
            WITH Cen DO
              BEGIN x:=x+Pi0.x; y:=y+Pi0.y; z:=z+Ht; END;
            END;
          FOR Ind:=1 TO 8 DO
            BEGIN
            CASE Ind OF
              1: BEGIN Xp:=x+1; Yp:=y;   END;
              2: BEGIN Xp:=x-1; Yp:=y;   END;
              3: BEGIN Xp:=x;   Yp:=y+1; END;
              4: BEGIN Xp:=x;   Yp:=y-1; END;
              5: BEGIN Xp:=x+1; Yp:=y+1; END;
              6: BEGIN Xp:=x+1; Yp:=y-1; END;
              7: BEGIN Xp:=x-1; Yp:=y+1; END;
              8: BEGIN Xp:=x-1; Yp:=y-1; END;
              END;
```

```
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              IF (TabSurfP[Xp,Yp]=IndSurfP) THEN
                BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
                Pi2.Suivant:=NIL; Pi1:=Pi2; TabSurfP[Xp,Yp]:=-IndSurf;
                WITH Sur[IndSurf] DO Taille:=Taille+1;
                IF (TabSurfC[Xp,Yp]<>1)AND(TabSeuil[Xp,Yp]<>0)THEN
                  TabSurfC[Xp,Yp]:=IndSurf;
                END;
              END;
            Debut:=Suivant;Dispose(Pi0);Pi0:=Debut;
            END;
        WITH Sur[IndSurf],Cen,Nor DO
         IF NbPt>=SeuilSurf THEN
          BEGIN z:=ROUND((z*CoefZ)/(100*NbPt))+100;
          x:=ROUND((x*CoefX)/NbPt); y:=ROUND((y*CoefY)/NbPt);
          Vxx:=Mxx-(Mx**2/NbPt); Vyy:=Myy-(My**2/NbPt); Vxy:=Mxy-((Mx*My)/NbPt);
          Vzz:=(Mzz-(Mz**2/NbPt))/10000; Vzx:=(Mzx-((Mz*Mx)/NbPt))/100;
          Vzy:=(Mzy-((Mz*My)/NbPt))/100;
          u:=-((Vzx*Vyy-Vzy*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefX);
          v:=-((Vzy*Vxx-Vzx*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefY);
          w:=1; NormerVec(Nor);
          END;
        END;
      END;
END;


{=============================================================================}
                      PROCEDURE SuppressionSurfaceFaible;
{=============================================================================}

BEGIN
FOR InSA1:=2 TO 200 DO
  IF (Sur[InSA1].NbPt<SeuilSurf) THEN
    WITH Sur[InSA1],Ori DO
      BEGIN NEW(Pi0); NEW(Pi1); Pi0.x:=x; Pi0.y:=y; Pi0.Suivant:=Pi1;
      Pi1.Suivant:=NIL; TabSurfP[I,J]:=0; NbPt:=0;
      WHILE Pi0.Suivant<>NIL DO
        WITH Pi0 DO
          BEGIN
          FOR Ind:=1 TO 8 DO
            BEGIN
            CASE Ind OF
              1: BEGIN Xp:=x+1; Yp:=y;   END;
              2: BEGIN Xp:=x-1; Yp:=y;   END;
```

```
              3: BEGIN Xp:=x;   Yp:=y+1; END;
              4: BEGIN Xp:=x;   Yp:=y-1; END;
              5: BEGIN Xp:=x+1; Yp:=y+1; END;
              6: BEGIN Xp:=x+1; Yp:=y-1; END;
              7: BEGIN Xp:=x-1; Yp:=y+1; END;
              8: BEGIN Xp:=x-1; Yp:=y-1; END;
              END;
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              IF (TabSurfP[Xp,Yp]=-InSA1) THEN
                BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
                Pi2.Suivant:=NIL; Pi1:=Pi2; TabSurfP[Xp,Yp]:=0;
                END;
            END;
          Debut:=Suivant;Dispose(Pi0);Pi0:=Debut;
          END;
      END;
END;

{=================================================================}
                  PROCEDURE RecherPointSautVertical;
{=================================================================}

VAR
  HtMin,HtMax,Ht                    : INTEGER;
  Vec                               : Vecteur;
  AlphaDeg,Dist                     : INTEGER;
  K,L,X,Y                           : INTEGER;
  Mx,Mxx,My,Myy,Mz,Mzz,Mxy,Mzx,Mzy : REAL;
  Vxx,Vyy,Vzz,Vxy,Vzx,Vzy           : REAL;

BEGIN
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabSurfC[I,J]=1 THEN
      BEGIN NbSurfAdj:=0; HtMin:=9999; HtMax:=0;
      FOR K:=-1 TO 1 DO
        FOR L:=-1 TO 1 DO
          IF ABS(TabSurfC[I+K,J+L])<>1 THEN
            BEGIN NouvelleSurf:=TRUE;IndSurf:=TabSurfC[I+K,J+L];
            FOR IndSurfAdj:=1 TO NbSurfAdj DO
              IF Adj[IndSurfAdj]=IndSurf THEN NouvelleSurf:=FALSE;
            IF (NouvelleSurf)AND(Sur[IndSurf].NbPt>0) THEN
              BEGIN NbSurfAdj:=NbSurfAdj+1; Adj[NbSurfAdj]:=IndSurf; END;
            END;
```

```
      FOR Ind1:=1 TO NbSurfAdj DO
        WITH Sur[Adj[Ind1]],Cen,Vec DO
          BEGIN u:=x-(I*CoefX); v:=y-(J*CoefY); w:=z;
          Ht :=ROUND(ProdScal(Vec,Nor)/Nor.w);
          IF Ht<HtMin THEN HtMin:=Ht; IF Ht>HtMax THEN HtMax:=Ht;
          END;
      IF HtMax-HtMin>SeuilSaut THEN
        BEGIN TabSurfC[I,J]:=-1;
        Mx:=0; Mxx:=0; My:=0; Myy:=0; Mz:=0; Mzz:=0; Mxy:=0; Mzx:=0; Mzy:=0;
        FOR K:=-1 TO 1 DO
          FOR L:=-1 TO 1 DO
            BEGIN  X:=I+K; Y:=J+L; Ht:=TabDist[X,Y];
            Mx:=Mx+X; Mxx:=Mxx+X*X; My:=My+Y; Myy:=Myy+Y*Y; Mz:=Mz+Ht;
            Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+X*Y; Mzx:=Mzx+Ht*X; Mzy:=Mzy+Ht*Y;
            END;
        Vxx:=Mxx-(Mx**2/9); Vyy:=Myy-(My**2/9); Vxy:=Mxy-((Mx*My)/9);
        Vzz:=(Mzz-(Mz**2/9))/10000; Vzx:=(Mzx-((Mz*Mx)/9))/100;
        Vzy:=(Mzy-((Mz*My)/9))/100;
        WITH Vec DO
          BEGIN
          u:=-((Vzx*Vyy-Vzy*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefX);
          v:=-((Vzy*Vxx-Vzx*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefY);
          w:=1; NormerVec(Vec); w:=0; NormerVec(Vec);
          IF u=0 THEN
            BEGIN Alpha:=3.1415926534/2; IF v<0 THEN Alpha:=-Alpha; END
          ELSE
            BEGIN Alpha:=ARCTAN(v/u);
            IF u<0 THEN
              BEGIN
              IF v>=0 THEN Alpha:=3.1415926534+Alpha
              ELSE Alpha:=-3.1415926534+Alpha;
              END;
            END;
          END;
        AlphaDeg:=ROUND(Alpha*Coef180Pi); TabDrOri[I,J]:=AlphaDeg;
        AlphaDeg:=AlphaDeg DIV 3; IF AlphaDeg=-60 THEN AlphaDeg:=60;
        Dist:=ROUND(Cos(Alpha)*I+Sin(Alpha)*J); Dist:=Dist DIV 3;
        TabDrDis[I,J]:=Dist; Accu[Dist,AlphaDeg]:=Accu[Dist,AlphaDeg]+1;
        END;
      END;
END;
```

```
{=============================================================================}
                        PROCEDURE ClassificationPoints;
{=============================================================================}

VAR
  K,L,JL,Somme,IndClasse,Coef : INTEGER;

BEGIN
FOR I:=-60 TO 60 DO
  FOR J:=-59 TO 60 DO
    BEGIN Somme:=0;
    FOR K:=-1 TO 1 DO
      FOR L:=-1 TO 1 DO
        BEGIN JL:=J+L; IF JL<-59 THEN JL:=120+JL; IF JL>60 THEN JL:=-120+JL;
        Somme:=Somme+Accu[I+K,JL];
        END;
    IF Somme > 5 THEN Classe[I,J]:=1;
    END;
IndClasse:=1;
FOR I:=-60 TO 60 DO
  FOR J:=-59 TO 60 DO
    IF Classe[I,J]=1 THEN
      BEGIN IndClasse:=IndClasse+1;
      NEW(Pi0); NEW(Pi1); Pi0.x:=I; Pi0.y:=J; Pi0.Suivant:=Pi1;
      Pi1.Suivant:=NIL; Classe[I,J]:=IndClasse;
      WHILE Pi0.Suivant<>NIL DO
        WITH Pi0 DO
          BEGIN
          FOR Ind:=1 TO 8 DO
            BEGIN
            CASE Ind OF
              1: BEGIN Xp:=x+1; Yp:=y;   END;
              2: BEGIN Xp:=x-1; Yp:=y;   END;
              3: BEGIN Xp:=x;   Yp:=y+1; END;
              4: BEGIN Xp:=x;   Yp:=y-1; END;
              5: BEGIN Xp:=x+1; Yp:=y+1; END;
              6: BEGIN Xp:=x+1; Yp:=y-1; END;
              7: BEGIN Xp:=x-1; Yp:=y+1; END;
              8: BEGIN Xp:=x-1; Yp:=y-1; END;
              END;
            IF(Xp>-60)AND(Xp<60)THEN
              BEGIN
              IF Yp<-59 THEN Yp:=120+Yp; IF Yp>60 THEN Yp:=Yp-120;
              IF Classe[Xp,Yp]=1 THEN
                BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
```

```
                Pi2.Suivant:=NIL; Pi1:=Pi2; Classe[Xp,Yp]:=IndClasse;
                END;
              END;
            END;
          Debut:=Suivant;Dispose(Pi0);Pi0:=Debut;
          END;
      END;
FOR K:=1 TO 2 DO
  BEGIN Coef:=(-1)**K;
  FOR I:=-60 TO 60 DO
    FOR J:=-59 TO 60 DO
      IF Coef*Classe[I,J]<0 THEN
        BEGIN Classe[I,J]:=-Classe[I,J];
        FOR Ind:=1 TO 8 DO
          BEGIN
          CASE Ind OF
            1: BEGIN Xp:=I+1; Yp:=J;   END;
            2: BEGIN Xp:=I-1; Yp:=J;   END;
            3: BEGIN Xp:=I;   Yp:=J+1; END;
            4: BEGIN Xp:=I;   Yp:=J-1; END;
            5: BEGIN Xp:=I+1; Yp:=J+1; END;
            6: BEGIN Xp:=I+1; Yp:=J-1; END;
            7: BEGIN Xp:=I-1; Yp:=J+1; END;
            8: BEGIN Xp:=I-1; Yp:=J-1; END;
            END;
          IF (Xp>-60)AND(Xp<60) THEN
            BEGIN IF Yp<-59 THEN Yp:=120-Yp; IF Yp>60 THEN Yp:=Yp-120;
            IF Classe[Xp,Yp]=0 THEN Classe[Xp,Yp]:=Classe[I,J];
            END;
          END;
        END;
  END;
END;


{=============================================================================}
                        PROCEDURE EtiquettePoints;
{=============================================================================}

BEGIN
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabSurfC[I,J]=-1 THEN
      IF Classe[TabDrDis[I,J],TabDrOri[I,J]DIV 3]<>0 THEN
        TabDrDis[I,J]:=-ABS(Classe[TabDrDis[I,J],TabDrOri[I,J]DIV 3])
```

```
      ELSE TabDrDis[I,J]:=0;
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabDrDis[I,J]<0 THEN
      BEGIN TabDrDis[I,J]:=-TabDrDis[I,J];
      FOR Ind:=1 TO 8 DO
        BEGIN
        CASE Ind OF
          1: BEGIN Xp:=I+1; Yp:=J;   END;
          2: BEGIN Xp:=I-1; Yp:=J;   END;
          3: BEGIN Xp:=I;   Yp:=J+1; END;
          4: BEGIN Xp:=I;   Yp:=J-1; END;
          5: BEGIN Xp:=I+1; Yp:=J+1; END;
          6: BEGIN Xp:=I+1; Yp:=J-1; END;
          7: BEGIN Xp:=I-1; Yp:=J+1; END;
          8: BEGIN Xp:=I-1; Yp:=J-1; END;
          END;
        IF (Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127) THEN
          IF TabDrDis[Xp,Yp]=0 THEN TabDrDis[Xp,Yp]:=TabDrDis[I,J];
        END;
      END;
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabSurfC[I,J]<>-1 THEN TabDrDis[I,J]:=0;
END;


{=================================================================}
                  PROCEDURE CaractVerticalesSurfaces;
{=================================================================}

VAR
  NbPoint,IndDr                        : INTEGER;
  Mx,Mxx,My,Myy,Mz,Mzz,Mxy,Mzx,Mzy : REAL;
  Vxx,Vyy,Vzz,Vxy,Vzx,Vzy              : REAL;
  Nb,Ht                                : INTEGER;

BEGIN NbSurfNV:=NbSurf;
FOR I:=2 TO 125 DO
  FOR J:=2 TO 125 DO
    IF TabDrDis[I,J]>1 THEN
      BEGIN NEW(Pi0); NEW(Pi1); Pi0.x:=I; Pi0.y:=J; Pi0.Suivant:=Pi1;
      Pi1.Suivant:=NIL; NbSurf:=NbSurf+1; IndDr:=TabDrDis[I,J];
      TabDrDis[I,J]:=-NbSurf; TabSurFC[I,J]:=-NbSurf;
      Sur[NbSurf].NumBloc:=TabSeuil[I,J]; Mx:=0; Mxx:=0; My:=0; Myy:=0;
```

```
Mz:=0; Mzz:=0; Mxy:=0; Mzx:=0; Mzy:=0; Nb:=0;
WHILE Pi0.Suivant<>NIL DO
  WITH Pi0,Sur[NbSurf] DO
    BEGIN
    WITH Cen,Nor DO
      BEGIN Ht:=TabDist[Pi0.x,Pi0.y]; NbPt:=NbPt+1; Nb:=Nb+1;
      Mx:=Mx+Pi0.x; Mxx:=Mxx+Pi0.x*Pi0.x;
      My:=My+Pi0.y; Myy:=Myy+Pi0.y*Pi0.y;
      Mz:=Mz+Ht; Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+Pi0.x*Pi0.y;
      Mzx:=Mzx+Ht*Pi0.x; Mzy:=Mzy+Ht*Pi0.y;
      x:=x+Pi0.x; y:=y+Pi0.y; z:=z+Ht;
      END;
    FOR Ind:=1 TO 24 DO
      BEGIN
      CASE Ind OF
        1 : BEGIN Xp:=x+1; Yp:=y;   END;
        2 : BEGIN Xp:=x-1; Yp:=y;   END;
        3 : BEGIN Xp:=x;   Yp:=y+1; END;
        4 : BEGIN Xp:=x;   Yp:=y-1; END;
        5 : BEGIN Xp:=x+1; Yp:=y+1; END;
        6 : BEGIN Xp:=x+1; Yp:=y-1; END;
        7 : BEGIN Xp:=x-1; Yp:=y+1; END;
        8 : BEGIN Xp:=x-1; Yp:=y-1; END;
        9 : BEGIN Xp:=x+2; Yp:=y;   END;
        10: BEGIN Xp:=x-2; Yp:=y;   END;
        11: BEGIN Xp:=x;   Yp:=y+2; END;
        12: BEGIN Xp:=x;   Yp:=y-2; END;
        13: BEGIN Xp:=x+2; Yp:=y+2; END;
        14: BEGIN Xp:=x+2; Yp:=y-2; END;
        15: BEGIN Xp:=x-2; Yp:=y+2; END;
        16: BEGIN Xp:=x-2; Yp:=y-2; END;
        17: BEGIN Xp:=x+2; Yp:=y+1  END;
        18: BEGIN Xp:=x+2; Yp:=y-1  END;
        19: BEGIN Xp:=x-2; Yp:=y+1; END;
        20: BEGIN Xp:=x-2; Yp:=y-1; END;
        21: BEGIN Xp:=x+1; Yp:=y+2; END;
        22: BEGIN Xp:=x-1; Yp:=y+2; END;
        23: BEGIN Xp:=x+1; Yp:=y-2; END;
        24: BEGIN Xp:=x-1; Yp:=y-2; END;
        END;
      IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
        BEGIN
        IF TabDrDis[Xp,Yp]=IndDr THEN
          BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
          Pi2.Suivant:=NIL; Pi1:=Pi2; TabDrDis[Xp,Yp]:=-NbSurf;
```

```
            TabSurFC[Xp,Yp]:=-NbSurf; Ht:=TabDist[Xp,Yp]; Nb:=Nb+1;
            Mx:=Mx+Xp; Mxx:=Mxx+Xp*Xp; My:=My+Yp; Myy:=Myy+Yp*Yp;Mz:=Mz+Ht;
            Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+Xp*Yp; Mzx:=Mzx+Ht*Xp; Mzy:=Mzy+Ht*Yp;
            END;
          IF (TabDrDis[Xp,Yp]=0)AND(Ind<9) THEN
            BEGIN TabDrDis[Xp,Yp]:=-1; Ht:=TabDist[Xp,Yp]; Nb:=Nb+1;
            Mx:=Mx+Xp; Mxx:=Mxx+Xp*Xp; My:=My+Yp; Myy:=Myy+Yp*Yp;Mz:=Mz+Ht;
            Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+Xp*Yp; Mzx:=Mzx+Ht*Xp; Mzy:=Mzy+Ht*Yp;
            END;
          END;
        END;
      Debut:=Suivant;Dispose(Pi0);Pi0:=Debut;
      END;
  IF Sur[NbSurf].NbPt<5 THEN
    BEGIN NEW(Pi0); NEW(Pi1); Pi0.x:=I; Pi0.y:=J; Pi0.Suivant:=Pi1;
    Pi1.Suivant:=NIL; TabDrDis[I,J]:=0; TabSurFC[I,J]:=-1;
    WHILE Pi0.Suivant<>NIL DO
      WITH Pi0 DO
        BEGIN
        FOR Ind:=1 TO 24 DO
          BEGIN
          CASE Ind OF
            1 : BEGIN Xp:=x+1; Yp:=y;   END;
            2 : BEGIN Xp:=x-1; Yp:=y;   END;
            3 : BEGIN Xp:=x;   Yp:=y+1; END;
            4 : BEGIN Xp:=x;   Yp:=y-1; END;
            5 : BEGIN Xp:=x+1; Yp:=y+1; END;
            6 : BEGIN Xp:=x+1; Yp:=y-1; END;
            7 : BEGIN Xp:=x-1; Yp:=y+1; END;
            8 : BEGIN Xp:=x-1; Yp:=y-1; END;
            9 : BEGIN Xp:=x+2; Yp:=y;   END;
            10: BEGIN Xp:=x-2; Yp:=y;   END;
            11: BEGIN Xp:=x;   Yp:=y+2; END;
            12: BEGIN Xp:=x;   Yp:=y-2; END;
            13: BEGIN Xp:=x+2; Yp:=y+2; END;
            14: BEGIN Xp:=x+2; Yp:=y-2; END;
            15: BEGIN Xp:=x-2; Yp:=y+2; END;
            16: BEGIN Xp:=x-2; Yp:=y-2; END;
            17: BEGIN Xp:=x+2; Yp:=y+1  END;
            18: BEGIN Xp:=x+2; Yp:=y-1  END;
            19: BEGIN Xp:=x-2; Yp:=y+1; END;
            20: BEGIN Xp:=x-2; Yp:=y-1; END;
            21: BEGIN Xp:=x+1; Yp:=y+2; END;
            22: BEGIN Xp:=x-1; Yp:=y+2; END;
            23: BEGIN Xp:=x+1; Yp:=y-2; END;
```

```
              24: BEGIN Xp:=x-1; Yp:=y-2; END;
              END;
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              IF TabDrDis[Xp,Yp]=-NbSurf THEN
                BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
                Pi2.Suivant:=NIL; Pi1:=Pi2;
                TabDrDis[Xp,Yp]:=0; TabSurFC[Xp,Yp]:=-1;
                END;
            END;
          Debut:=Suivant;Dispose(Pi0);Pi0:=Debut;
          END;
      END
    ELSE
      WITH Sur[NbSurf],Cen,Nor DO
        BEGIN x:=ROUND((x*CoefX)/NbPt); y:=ROUND((y*CoefY)/NbPt);
        z:=ROUND((z*CoefZ)/(100*NbPt))+100;
        Vxx:=Mxx-(Mx**2/Nb); Vyy:=Myy-(My**2/Nb); Vxy:=Mxy-((Mx*My)/Nb);
        Vzz:=(Mzz-(Mz**2/Nb))/10000; Vzx:=(Mzx-((Mz*Mx)/Nb))/100;
        Vzy:=(Mzy-((Mz*My)/Nb))/100;
        u:=-((Vzx*Vyy-Vzy*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefX);
        v:=-((Vzy*Vxx-Vzx*Vxy)*CoefZ)/((Vxx*Vyy-Vxy**2)*CoefY);
        w:=1; NormerVec(Nor); w:=0; NormerVec(Nor); Taille:=NbPt;
        TabDrDis[ROUND(x/CoefX),ROUND(y/CoefY)]:=1;
        END;
    END;
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF TabDrDis[I,J]=-1 THEN TabDrDis[I,J]:=0;
END;


{=====================================================================}
                  PROCEDURE ConnectionEntreSurfaces;
{=====================================================================}

VAR
  HtMin,HtMax,Ht                : INTEGER;
  Vec                           : Vecteur;
  AlphaDeg,Dist,HGrad,VGrad     : INTEGER;
  K,L,JL,Somme,IndClasse,Coef : INTEGER;

BEGIN
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    IF (TabSurfC[I,J]=1)OR(TabSurfC[I,J]<-1) THEN
```

```
      BEGIN NbSurfAdj:=0;
      FOR K:=-1 TO 1 DO
        FOR L:=-1 TO 1 DO
          IF TabSurfC[I+K,J+L]>1 THEN
            BEGIN NouvelleSurf:=TRUE;IndSurf:=TabSurfC[I+K,J+L];
            IF IndSurf<>IndSol THEN
              BEGIN
              FOR IndSurfAdj:=1 TO NbSurfAdj DO
                IF Adj[IndSurfAdj]=IndSurf THEN NouvelleSurf:=FALSE;
              IF (NouvelleSurf)AND(Sur[IndSurf].NbPt>0) THEN
                BEGIN NbSurfAdj:=NbSurfAdj+1; Adj[NbSurfAdj]:=IndSurf; END;
              END;
            END;
      IF TabSurfC[I,J]=1 THEN
        FOR Ind1:=1 TO NbSurfAdj-1 DO
          FOR Ind2:=Ind1+1 TO NbSurfAdj DO
            BEGIN InSA1:=Adj[Ind1]; InSA2:=Adj[Ind2];
            TAdj[InSA1,InSA2]:=TRUE; TAdj[InSA1,1]:=TRUE;
            TAdj[InSA2,InSA1]:=TRUE; TAdj[InSA2,1]:=TRUE;
            END
      ELSE
        FOR Ind1:=1 TO NbSurfAdj DO
          BEGIN InSA1:=-TabSurfC[I,J]; InSA2:=Adj[Ind1];
          TAdj[InSA1,InSA2]:=TRUE; TAdj[InSA1,1]:=TRUE;
          TAdj[InSA2,InSA1]:=TRUE; TAdj[InSA2,1]:=TRUE;
          END;
      END;
END;


{=============================================================================}
                  PROCEDURE EcritCoupleSurfacesAdjacentes;
{=============================================================================}

 VAR
   Vect : Vecteur;

BEGIN
FOR IndBloc:=2 TO NbBloc DO
  BEGIN NbCSA:=0;
  WRITELN(' Bloc numero : ',IndBloc);WRITELN;
  FOR InSA1:=2 TO 200 DO
    IF (TAdj[InSA1,1])AND(Sur[InSA1].NumBloc=IndBloc) THEN
      BEGIN TAdj[InSA1,1]:=FALSE;
      WRITE('Surf ',InSA1:4,' : ');
```

```
      FOR InSA2:=2 TO 200 DO
        IF TAdj[InSA1,InSA2] THEN
          BEGIN NbCSA:=NbCSA+1;
          WITH SurIm DO
            BEGIN InBloc:=IndBloc;
            WITH Sur[InSA1] DO
              BEGIN I1:=InSA1; P1:=Cen; N1:=Nor; S1:=Taille; END;
            WITH Sur[InSA2] DO
              BEGIN I2:=InSA2; P2:=Cen; N2:=Nor; S2:=Taille; END;
            WITH Vect,P1 DO BEGIN u:=x-P2.x; v:=y-P2.y; w:=z-P2.z; END;
            IF ProdScal(N2,Vect)>0 THEN ConvIm:=FALSE ELSE ConvIm:=TRUE;
            DisIm:=CarreDistPoint(P1,P2);
            AngIm:=ArcCos(ProdScal(N1,N2));
            WRITE(InSA2:4,'(',AngIm:3);
            IF ConvIm THEN WRITE('+).') ELSE WRITE('-).');
            END;
          WRITE(FicSurIm,SurIm);
          END;
      WRITELN;
      END;
  WRITELN;WRITELN(' Nombre de couples de surfaces adjacentes : ',NbCSA);
  WRITELN;WRITELN;
  END;
END;


{-------------------------------------------------------------------------}
                          PROCEDURE SauveImages;
{-------------------------------------------------------------------------}

BEGIN
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO
    BEGIN
    IF TabSurfP[I,J]<0 THEN TabSurfP[I,J]:=-TabSurfP[I,J]
    ELSE TabSurfP[I,J]:=0;
    IF TabDrDis[I,J]<0 THEN TabDrDis[I,J]:=-TabDrDis[I,J];
    IF TabSeuil[q,J]<>0 THEN TabDist[I,J]:=TabDist[I,J]+10000
    ELSE TabDist[I,J]:=0;
    END;
EcritureImage(FicSurEt,TabSurfP); EcritureImage(FicSurfV,TabDrDis);
EcritureImage(FicDist,TabDist);
END;
```

```
{*************************************************************************}
                          PROGRAMME PRINCIPAL
{*************************************************************************}

BEGIN
Initialisation;
RecherIndiceSol;
CaractPlanesSurfaces;
SuppressionSurfaceFaible;
RecherPointSautVertical;
ClassificationPoints;
EtiquettePoints;
CaractVerticalesSurfaces;
ConnectionEntreSurfaces;
EcritCoupleSurfacesAdjacentes;
SauveImages;
END.
```

G. "ANALYSESCENE" PROGRAM

This program is used to make the complete analysis of the studied scene (i.e., recognition of each polyhedron present in the scene and precise determination of its position).

Parameters

- Name of models. This name corresponds only to the "prefix" of the CAD or LGP-Models name which must be agree with the format [prefix + number . extension].

- Total number of different models must be used during the analysis. The program reads automatically these models and for this reason the number of the models must vary continuously from 1 to N (N is equal to the total number of different models used).

- Name of the scene studied. This name is not followed by any extension (e.g., IMAGE02).

- Allowed angular variation for the matching of scene and model pairs of adjacent surfaces.

- Three thresholds used by the clustering program to find the model view axis, the model view orientations, and the model centers.

- Millimetre distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis) and between two levels of the range data (Z-axis).

- Number of the group of overlapping objects which must be studied. This number can be read on the image showing the location of these groups (see Fig. A-6a). If the number 0 is entered, all the groups of the scene are studied.

- Number of a particular model only if the occurrence of this model in the scene must be searched. If the number 0 is entered, all models are used.

Input Files

- File containing the characteristics of each pair of adjacent plane surfaces of the studied scene (for more details see paragraph II.2).

- Location of the different groups of overlapping objects (see Fig. A-6a). This location corresponds to the image obtained by thresholding the range data.

- New labelling of the plane surfaces after cancellation of the too small ones (see Fig. A-6b).

- Modified range data (see Fig. A-6c). The distance between the background and the objects is increased to penalise the bad matches between model and scene object (see paragraph II.7).

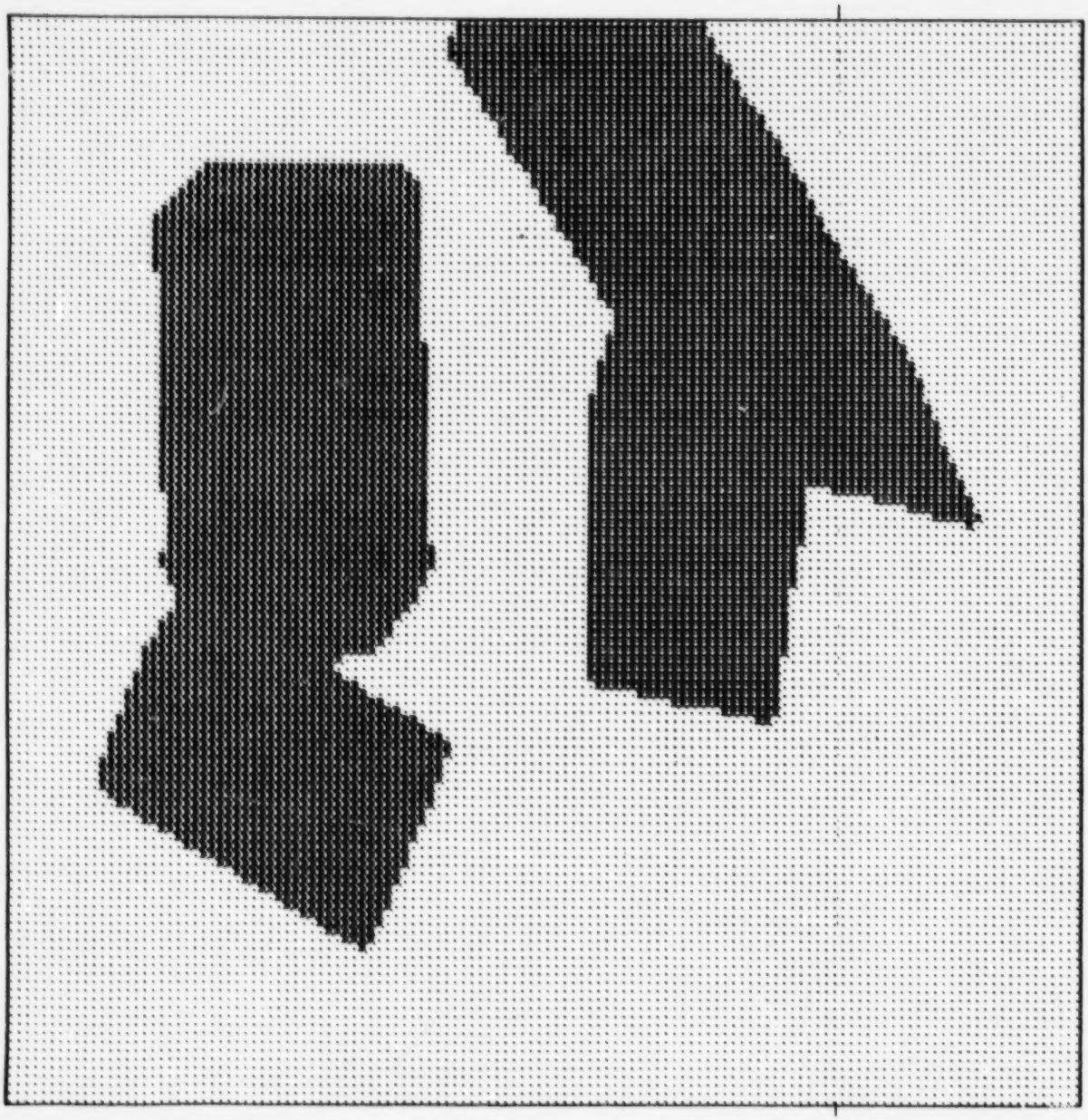
Note: These four files are automatically read by the program. For this reason, the name of these files must be equal to the name of the studied image followed respectively by the extensions ".CSA", ".SEU", ".SET" and ".DIM" (e.g., IMAGE02.CSA).

- CAD-Model of each polyhedron can be found in the studied scene.

- LGP-Model of each polyhedron can be found in the studied scene.

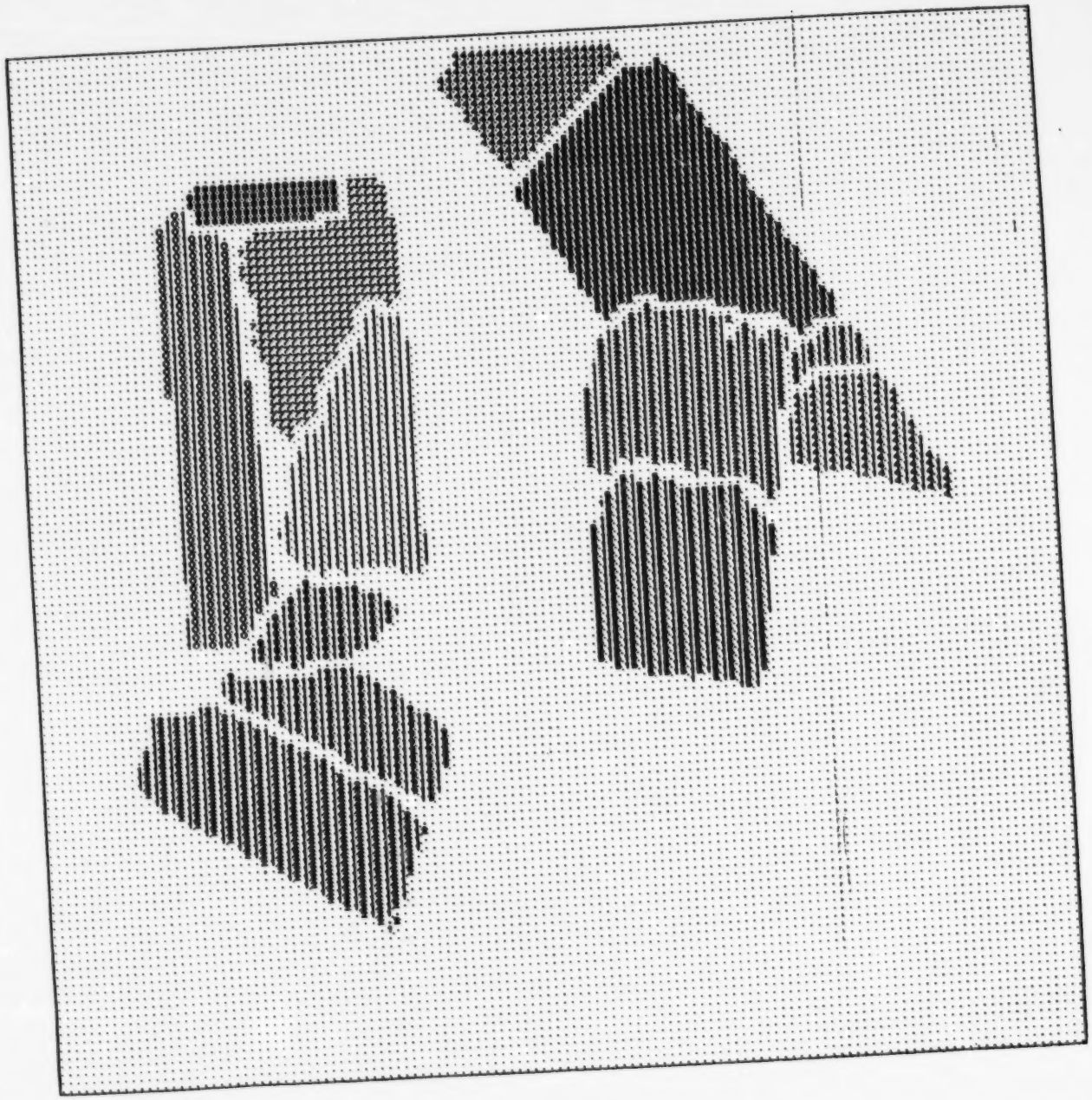
Output File

- Text file containing the result of the analysis (see Fig. A-7). The name of this file is the same as the one of the studied image followed by the extension ".SOL" (e.g., IMAGE02.SOL).



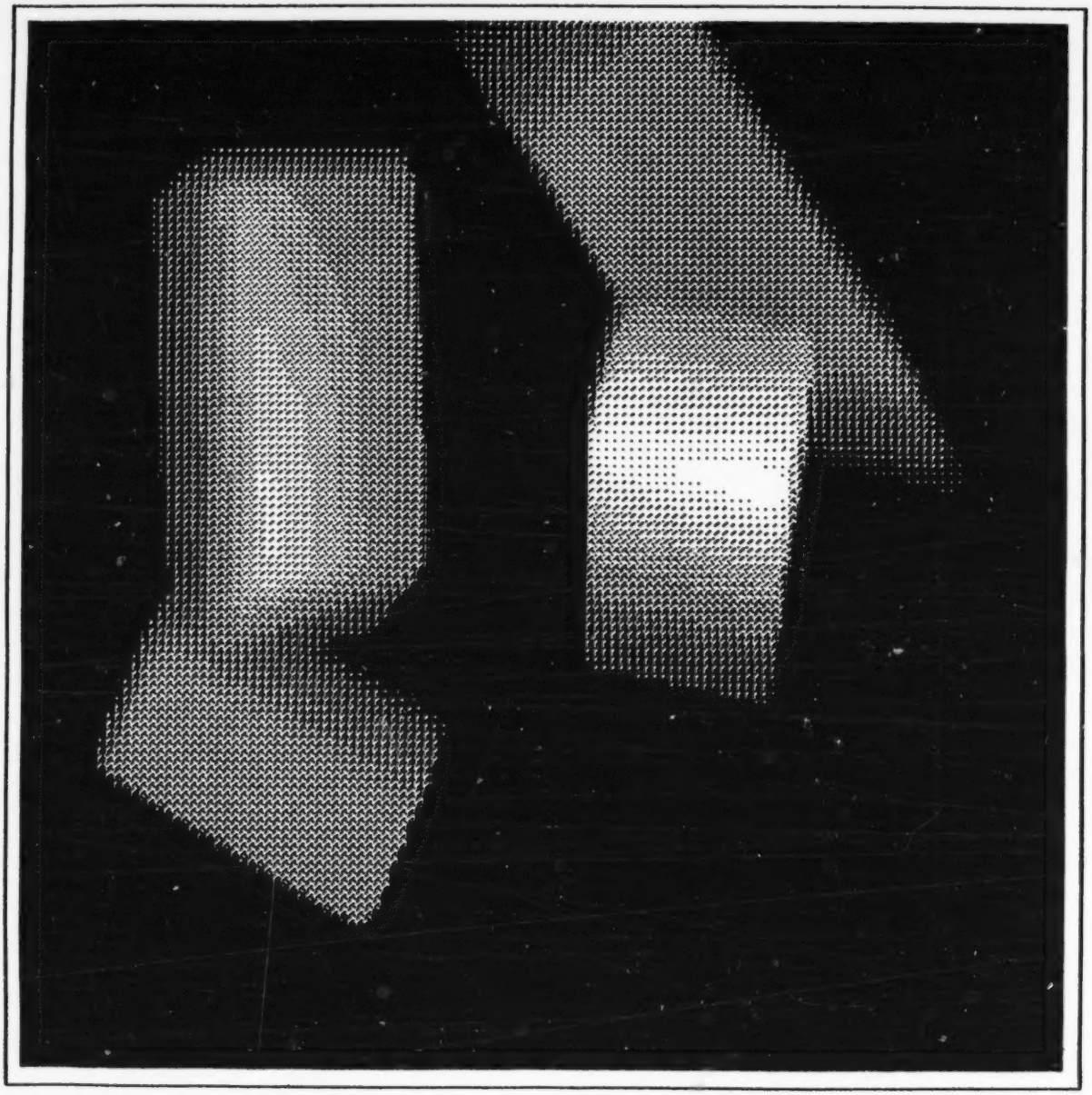
(a)

Figure A-6. Input files of the "ANALYSESCENE" program. (a) Location of the different groups of overlapping objects. (b) New labelling of the plane surfaces. (c) Modified range data.



(b)

Figure A-6. (Continued).



(c)

Figure A-6. (Concluded).

**** RESULTATS DE l'ANALYSE ****

```
Nom des objets ? MODEL
Nombre d'objets possibles ? 11
Nom de l'image a traiter ? IMAGE05
Valeur angulaire permise (degre) ? 10
Seuils clustering (degre,degre,mm) ? 15  12  6
CoefX,CoefY,CoefZ ?  7.20E-01  8.66E-01  1.00E+00
Indice bloc a traiter (tous=0) ? 0
Indice Objet a rechercher (tous=0) ? 0
```

********** 2 Bloc **********

No Modele	Position Empilement	Rotation R_x	R_y	R_z	Translation T_x	T_y	T_z	Indice qualite (10E-05 M)
1	2	0	0	-52	19	71	9	82
5	1	-129	-2	79	42	71	15	41

********** 3 Bloc **********

No Modele	Position Empilement	Rotation R_x	R_y	R_z	Translation T_x	T_y	T_z	Indice qualite (10E-05 M)
6	2	-176	-89	151	66	15	3	32
9	1	174	-57	97	26	22	13	99

```
TEMPS CPU      709030 = 00:11:49.03
```

Figure A-7. Result of the analysis of the scene corresponding to Fig. A-6.

```
{*****************************************************************************}
                    PROGRAM AnalyseScene(INPUT,OUTPUT);
{*****************************************************************************}


{========================== DECLARATION DE TYPES ===========================}

TYPE
  Degre       = -180..180;
  Bytes       = -32766..32765;
  Ligne       = PACKED ARRAY[0..127] OF Bytes;
  Plan        =  ARRAY[0..127] OF ligne;
  FichierPlan = FILE OF Ligne;
  Point = RECORD
        x,y,z         : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w         : REAL;
        END;
  Rotation = RECORD
        a,b,g         : INTEGER;
        END;
  ElementSurface = RECORD
        x,y           : INTEGER;
        Suivant       : ElementSurface;
        END;
  Surface = RECORD
        Nor           : Vecteur;
        Cen           : Point;
        Taille        : INTEGER;
        NbSomLie      : 1..20;
        SomLie        : ARRAY[1..20] OF 1..100;
        NbSurAdj      : 1..20;
        SurAdj        : ARRAY[1..20] OF 1..20;
        END;
  Objet = RECORD
        NbSom         : 1..100;
        Som           : ARRAY[1..100] OF Point;
        NbSur         : 1..20;
        Sur           : ARRAY[1..20] OF Surface;
        END;
  PairSurIm = RECORD
        AngIm         : Degre;
        DisIm         : INTEGER;
        InBloc        : INTEGER;
```

```
        ConvIm        : BOOLEAN;
        Ii1,Ii2       : INTEGER;
        P1,P2         : Point;
        Ni1,Ni2       : Vecteur;
        Si1,Si2       : INTEGER;
        END;
RecPSurIm = RECORD
        Valeur        : PairSurIm;
        Suivant       : RecPSurIm;
        END;
PairSurMod = RECORD
        AngMod        : Degre;
        DisMod        : INTEGER;
        ConvMod       : BOOLEAN;
        Im1,Im2       : 1..20;
        D1,D2         : REAL;
        Nm1,Nm2       : Vecteur;
        Sm1,Sm2       : INTEGER;
        END;
RecPSurMod = RECORD
        Valeur        : PairSurMod;
        Suivant       : RecPSurMod;
        END;
PairSurCor = RECORD
        PSIm          : RecPSurIm;
        PSMod         : RecPSurMod;
        CoefSurf      : INTEGER;
        END;
AxeVue = RECORD
        VecAxe,N1,N2 : Vecteur;
        IndAxeV       : 1..20;
        END;
OrientVue = RECORD
        VecOri        : Vecteur;
        IndVOri       : 1..5;
        IndAV         : 1..600;
        Vec           : Vecteur;
        Poi           : Point;
        END;
Centre = RECORD
        Cen           : Point;
        IndR1,IndR2   : 1..100;
        IndTr         : 1..5;
        END;
TabSurfConnecte = PACKED ARRAY[1..20,0..20] OF 1..255;
```

```
  CorObjSurf = RECORD
        IndCl          : INTEGER;
        IndOb          : INTEGER;
        Rot            : Rotation;
        Tra            : Vecteur;
        NbCen          : INTEGER;
        DifH           : INTEGER;
        SurOb          : INTEGER;
        SurObBc        : INTEGER;
        TabSuCo        : TabSurfConnecte;
        END;
  TabPairSurCor    = ARRAY[1..600] OF PairSurCor;
  TabAxeVue        = ARRAY[1..600] OF AxeVue;
  TabOrientVue     = ARRAY[1..100] OF OrientVue;
  TabCentre        = ARRAY[1..100] OF Centre;
  TabSurfaceBloc   = ARRAY[0..20] OF INTEGER;
  TabCorObjSurf    = ARRAY[2..20,1..20,1..10] OF CorObjSurf;
  TabCombSurface   = ARRAY[2..20,1..20,1..10] OF PACKED SET OF 1..100;
  Nom              = VARYING[12] OF CHAR;

{====================== DECLARATION DE VARIABLES ======================}

VAR
  FicModele                              : FILE OF PairSurMod;
  FicImage                               : FILE OF PairSurIm;
  FicDis,FicBloc,FicSurfP                : FichierPlan;
  FicText                                : TEXT;
  NomObj,NomFicObj,NomFicMod             : Nom;
  NomIma,NomFicIm,NomFicBloc             : Nom;
  NomFicDis,NomFicSuP,Num,NomFicText     : Nom;
  PSurMod,PtMod                          : RecPSurMod;
  PSurIm,PtIm                            : RecPSurIm;
  Seuil1,Seuil2,Seuil3,Seuil4            : REAL;
  Obj                                    : Objet;
  IndSur,IndSom,IndSomLie                : 1..100;
  FicObjet                               : FILE OF Objet;
  Rotat                                  : Rotation;
  Transl                                 : Vecteur;
  NbCentre                               : INTEGER;
  DifHt,NbPiOb,NbPiSuOb,NbPiObBc         : INTEGER;
  TabDist,TabBloc,TabSurfP,TabHtOb       : Plan;
  HeureDebut                             : INTEGER;
  IndBloc,NbBloc                         : INTEGER;
  IndObj,NbObj                           : INTEGER;
  TabPSC                                 : TabPairSurCor;
```

```
  TabAxV                                  : TabAxeVue;
  TabOri                                  : TabOrientVue;
  TabCen                                  : TabCentre;
  TabSC                                   : TabSurfConnecte;
  TabPiBc                                 : TabSurfaceBloc;
  TabCObSu                                : TabCorObjSurf;
  PtSur                                   : ARRAY[1..200] OF Point;
  IndAxV,NbAxV,IndPairCor,NbPairCor       : 1..600;
  IndOriVu,NbOriVu,NbVecOri               : 1..100;
  IndTransf,NbTransf,NbCen                : 1..100;
  IndBlocExa,IndObjExa,SurfBloc           : INTEGER;
  CoefX,CoefY,CoefZ                       : REAL;
  IndObjet,IndClasse                      : INTEGER;
  Pi0,Pi1,Pi2,Debut                       : ElementSurface;
  P1,P2                                   : Point;
  Ind,Ind1,Xp,Yp,I,J,K                    : INTEGER;
  ComSurf                                 : PACKED SET OF 1..100;
  TComSurf                                : TabCombSurface;
  NbComSurf                               : ARRAY[2..20,1..20] OF INTEGER;
  IndComSurf,IndComSurf1                  : INTEGER;

{========================= DECLARATION DE PROCEDURES =========================}

FUNCTION  ProdScal(V1,V2 : Vecteur):REAL;                                EXTERN;
PROCEDURE ProdVect(V1,V2 : Vecteur; VAR V0 : Vecteur);                   EXTERN;
PROCEDURE ProdVectNor(V1,V2 : Vecteur; VAR V0 : Vecteur);                EXTERN;
FUNCTION  ArcCos(Alpha : REAL) : Degre;                                  EXTERN;
PROCEDURE NormerVec(VAR V : Vecteur);                                    EXTERN;
PROCEDURE Translation(Tra : Vecteur; Var P : Point);                     EXTERN;
PROCEDURE RotationVect(Rot : Rotation; VAR Vec : Vecteur);               EXTERN;
PROCEDURE RotationPoint(Rot : Rotation; VAR P : Point);                  EXTERN;
PROCEDURE AxeDeVue(Vi1,Vi2 : Vecteur; Var Vf1,Vf2,VAxe: Vecteur);        EXTERN;
PROCEDURE OrientationVue(Vi1,Vi2 : Vecteur;
                                    Var Vf1,Vf2,VAxe: Vecteur);          EXTERN;
PROCEDURE PointDroiteInter(P1,P2 : Point; D1,D2 : REAL;
                           V1,V2 : Vecteur; VAR P0 : Point);             EXTERN;
PROCEDURE MilieuCoupleDroite(SeuilDis,LargOb: INTEGER; P1,P2:Point;
                             N1,N2:Vecteur; VAR P0 :Point);              EXTERN;
PROCEDURE LectureImage(VAR Fic : FichierPlan; Tab : Plan);               EXTERN;
PROCEDURE EcritureImage(VAR Fic : FichierPlan; Tab : Plan);              EXTERN;
PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER; VAR Tab:PLAN);       EXTERN;
```

```
{=====================================================================}
          PROCEDURE Affinage( NbIter : INTEGER; VAR Tab : Plan);
{=====================================================================}

VAR
  A,B,C     : ARRAY[1..9] OF INTEGER;
  NbModif,M : INTEGER;
  X,X1,X2   : INTEGER;
  U,V,W     : INTEGER;

PROCEDURE Connectivite8(VAR X : INTEGER);

VAR
  K,Z : INTEGER;
  B   : ARRAY[1..4] OF INTEGER;

BEGIN X:=0;
FOR K:=1 TO 4 DO
  BEGIN Z:=2*K-1;
  B[K]:=(1-A[Z])-((1-A[Z])*(1-A[Z+1])*(1-A[Z+2])); X:=X+B[K];
  END;
END;


BEGIN M:=1;
FOR I:=0 TO 127 DO
  BEGIN Tab[I,0]:=-1; Tab[I,127]:=-1; Tab[0,I]:=-1; Tab[127,I]:=-1; END;
REPEAT NbModif:=0;
  FOR I:=1 TO 126 DO
    FOR J:=1 TO 126 DO
      IF Tab[I,J]>0 THEN
        BEGIN A[1]:=Tab[I,J+1]; A[2]:=Tab[I-1,J+1]; A[3]:=Tab[I-1,J];
        A[4]:=Tab[I-1,J-1]; A[5]:=Tab[I,J-1]; A[6]:=Tab[I+1,J-1];
        A[7]:=Tab[I+1,J]; A[8]:=Tab[I+1,J+1]; B:=A; C:=A;
        FOR K:=1 TO 8 DO
          IF A[K]>0 THEN BEGIN A[K]:=1; C[K]:=1; END
          ELSE
            IF A[K]=-M THEN BEGIN A[K]:=1; C[K]:=0; END
            ELSE IF A[K]<0 THEN BEGIN A[K]:=0; C[K]:=0; END;
        U:=4-A[1]-A[3]-A[5]-A[7]; A[9]:=A[1];
        IF U>0 THEN
          BEGIN V:=0; W:=0;
          FOR K:=1 TO 8 DO
            BEGIN IF A[K]=1 THEN V:=V+1; IF C[K]=1 THEN W:=W+1; END;
          IF (V>1) AND (W>0) THEN
```

```
            BEGIN Connectivite8(X);
            IF X=1 THEN
              BEGIN X1:=1; X2:=1;
              IF B[3]=-M THEN BEGIN A[3]:=0; Connectivite8(X1); A[3]:=1; END;
              IF B[5]=-M THEN BEGIN A[5]:=0; Connectivite8(X2); END;
              IF (X1=1) AND (X2=1) THEN
                BEGIN Tab[I,J]:=-M; NbModif:=NbModif+1; END;
              END;
            END;
          END;
        END;
  M:=M+1;
UNTIL (NbModif=0) OR (M=NbIter+1);
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO IF Tab[I,J]<0 THEN Tab[I,J]:=0;
END;


{=========================================================================}
            PROCEDURE Extension(NbIter : INTEGER; VAR Tab : Plan);
{=========================================================================}

VAR
  TabNul,TabMod1,TabMod2            : PACKED ARRAY[0..127,0..127] OF BOOLEAN;
  Mx,Mxx,My,Myy,Mz,Mzz,Mxy,Mzx,Mzy : REAL;
  Vxx,Vyy,Vzz,Vxy,Vzx,Vzy           : REAL;
  Ht,L,M,x,y,NbPt                   : INTEGER;
  Nor,Vec                           : Vecteur;

BEGIN
FOR J:=0 TO 127 DO TabNul[0,J]:=FALSE;
FOR I:=1 TO 127 DO TabNul[I]:=TabNul[0]; TabMod2:=TabNul;
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO IF Tab[I,J]<>0 THEN TabMod2[I,J]:=TRUE;
FOR K:=1 TO NbIter DO
  BEGIN TabMod1:=TabMod2; TabMod2:=TabNul;
  FOR I:=1 TO 126 DO
    FOR J:=1 TO 126 DO
      IF TabMod1[I,J] THEN
        BEGIN NbPt:=0;
        Mx:=0; Mxx:=0; My:=0; Myy:=0; Mz:=0; Mzz:=0; Mxy:=0; Mzx:=0; Mzy:=0;
        FOR L:=-1 TO 1 DO
          FOR M:=-1 TO 1 DO
            BEGIN x:=I+L; y:=J+M;
            IF Tab[x,y]<>0 THEN
```

```
            BEGIN NbPt:=NbPt+1; Ht:=Tab[x,y];
            Mx:=Mx+x; Mxx:=Mxx+x*x; My:=My+y; Myy:=Myy+y*y; Mz:=Mz+Ht;
            Mzz:=Mzz+Ht*Ht; Mxy:=Mxy+x*y; Mzx:=Mzx+Ht*x; Mzy:=Mzy+Ht*y;
            END;
          END;
        Vxx:=Mxx-(Mx**2/NbPt); Vyy:=Myy-(My**2/NbPt); Vxy:=Mxy-(Mx*My)/NbPt;
        Vzz:=Mzz-(Mz**2/NbPt); Vzx:=Mzx-(Mz*Mx)/NbPt; Vzy:=Mzy-(Mz*My)/NbPt;
        IF Vxx*Vyy-Vxy**2<>0 THEN
          BEGIN
          Nor.u:=-(Vzx*Vyy-Vzy*Vxy)/(Vxx*Vyy-Vxy**2);
          Nor.v:=-(Vzy*Vxx-Vzx*Vxy)/(Vxx*Vyy-Vxy**2);
          Nor.w:=1; NormerVec(Nor);
          END
        ELSE BEGIN Nor.u:=0; Nor.v:=0; Nor.w:=1; END;
        FOR Ind:=1 TO 4 DO
          BEGIN
          CASE Ind OF
            1: BEGIN Xp:=I+1; Yp:=J;   END;
            2: BEGIN Xp:=I-1; Yp:=J;   END;
            3: BEGIN Xp:=I;   Yp:=J+1; END;
            4: BEGIN Xp:=I;   Yp:=J-1; END;
            END;
          IF Tab[Xp,Yp]=0 THEN
            WITH Vec DO
              BEGIN u:=(I-Xp); v:=(J-Yp); w:=Tab[I,J]; TabMod2[Xp,Yp]:=TRUE;
              Tab[Xp,Yp]:=ROUND(ProdScal(Vec,Nor)/Nor.w);
              END;
          END;
        END;
  END;
END;


{=============================================================================}
                          PROCEDURE Initialisation;
{=============================================================================}

BEGIN
WRITELN(                          **** Programme d  analyse ****  );
WRITELN;WRITELN;
WRITE(  Nom des objets ?  );READLN(NomObj);WRITELN;
WRITE(  Nombre d  objets possibles ?  );READLN(NbObj);WRITELN;
WRITE(  Nom de l  image a traiter ?  );READLN(NomIma);WRITELN;
WRITE(  Valeur angulaire permise pour mise en correspondance (degre) ?  );
READLN(Seuil1);WRITELN;
```

```
WRITE(' Valeurs seuils pour clustering (degre,degre,mm) ? ');
READLN(Seuil2,Seuil3,Seuil4);WRITELN;
WRITE(' CoefX,CoefY,CoefZ ? ');READLN(CoefX,CoefY,CoefZ);WRITELN;
WRITE(' Indice bloc a traiter (tous=0) ? ');READLN(IndBlocExa);WRITELN;
WRITE(' Indice Objet a rechercher (tous=0) ? '); READLN(IndObjExa);WRITELN;
WRITELN;WRITELN;WRITELN;
NomFicIm:=NomIma+'.CSA';OPEN(FicImage,NomFicIm,OLD);RESET(FicImage);
NomFicDis:=NomIma+'.DIM';OPEN(FicDis,NomFicDis,OLD);
LectureImage(FicDis,TabDist);
NomFicBloc:=NomIma+'.SEU';OPEN(FicBloc,NomFicBloc,OLD);
LectureImage(FicBloc,TabBloc);
NomFicSuP:=NomIma+'.SET';OPEN(FicSurfP,NomFicSuP,OLD);
LectureImage(FicSurfP,TabSurfP);
NomFicText:=NomIma+'.SOL';OPEN(FicText,NomFicText,NEW);REWRITE(FicText);
WRITELN(FicText,'                          **** Programme d''analyse ****');
WRITELN(FicText);WRITELN(FicText);
WRITELN(FicText,' Nom des objets ? ',NomObj);
WRITELN(FicText,' Nombre d''objets possibles ? ',NbObj);
WRITELN(FicText,' Nom de l''image a traiter ? ',NomIma);
WRITELN(FicText,' Valeur angulaire permise (degre) ? ',ROUND(Seuil1));
WRITELN(FicText,' Seuils clustering (degre,degre,mm) ? ',ROUND(Seuil2),
              ROUND(Seuil3),ROUND(Seuil4));
WRITELN(FicText,' CoefX,CoefY,CoefZ ? ',CoefX,CoefY,CoefZ);
WRITELN(FicText,' Indice bloc a traiter (tous=0) ? ',IndBlocExa);
WRITELN(FicText,' Indice Objet a rechercher (tous=0) ? ',IndObjExa);
WRITELN(FicText);WRITELN(FicText);WRITELN(FicText);
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO
    IF TabSurfP[I,J]<>0 THEN
      IF PtSur[TabSurfP[I,J]].x=0 THEN
        WITH PtSur[TabSurfP[I,J]] DO
          BEGIN x:=I; y:=J; z:=0; END;
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO
    BEGIN TabPiBc[TabBloc[I,J]]:=TabPiBc[TabBloc[I,J]]+1; END;
FOR IndBloc:=2 TO NbBloc DO
  FOR IndObj:=1 TO NbObj DO NbComSurf[IndBloc,IndObj]:=0;
Affinage(3,TabDist); Extension(6,TabDist);
Seuil2:=COS((Seuil2*3.1415926534)/180);
Seuil3:=COS((Seuil3*3.1415926534)/180);
Seuil4:=1-(Seuil4/127);
HeureDebut:=CLOCK;
END;
```

```
{=========================================================================}
                    PROCEDURE LectureSceneEtudiee;
{=========================================================================}

BEGIN PtIm:=NIL;
WHILE NOT EOF(FicImage) DO
  BEGIN NEW(PSurIm);
  WITH PSurIm DO
    BEGIN READ(FicImage,Valeur); Suivant:=PtIm; PtIm:=PSurIm; END;
  END;
NbBloc:=PSurIM.Valeur.InBloc;
END;


{=========================================================================}
                    PROCEDURE LectureModeleObjet;
{=========================================================================}

BEGIN
IF (IndObj>1)OR(IndBloc<NbBloc) THEN
  BEGIN CLOSE(FicObjet); CLOSE(FicModele); END;
IF IndObj > 9 THEN Num:=CHR((IndObj DIV 10)+48)+CHR((IndObj MOD 10)+48)
ELSE Num:=CHR(IndObj+48);
NomFicObj:=NomObj+Num+'.CAD';
OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
NomFicMod:=NomObj+Num+'.LGP';
OPEN(FicModele,NomFicMod,OLD);RESET(FicModele);PtMod:=NIL;
WHILE NOT EOF(FicModele) DO
  BEGIN NEW(PSurMod);
  WITH PSurMod DO
    BEGIN READ(FicModele,Valeur); Suivant:=PtMod; PtMod:=PSurMod; END;
  END;
SurfBloc:=TabPiBc[IndBloc]; IndClasse:=0;
END;


{=========================================================================}
                  PROCEDURE MiseCorrespondanceSurfaces;
{=========================================================================}

VAR
  Coef1,Coef2 : INTEGER;

BEGIN PSurIm:=PtIm; NbPairCor:=0;
REPEAT
  PSurMod:=PtMod;
```

```
  REPEAT
    WITH PSurMod.Valeur,PSurIm.Valeur DO
      BEGIN
      IF (ConvMod=ConvIm) AND ((ABS(AngMod-AngIm)<=Seuil1)OR(AngIm=-1))
                                                  AND (DisIm<=DisMod) THEN
        BEGIN
        IF Ni1.w<>0 THEN Coef1:=ROUND((Si1*CoefX*CoefY*100)/(Ni1.w*Sm1))
        ELSE Coef1:=ROUND((Si1*CoefX*CoefY*100)/Sm1);
        IF Ni2.w<>0 THEN Coef2:=ROUND((Si2*CoefX*CoefY*100)/(Ni2.w*Sm2))
        ELSE Coef2:=ROUND((Si2*CoefX*CoefY*100)/Sm2);
        IF Coef1=0 THEN Coef1:=1; IF Coef2=0 THEN Coef2:=1;
        IF (Coef1<=150)AND(Coef2<=150) THEN
          BEGIN NbPairCor:=NbPairCor+1;
          WITH TabPSC[NbPairCor] DO
            BEGIN PSim:=PSurIm;PSMod:=PSurMod;CoefSurf:=Si1+Si2;END;
          END;
        END;
      END;
    PSurMod:=PSurMod.Suivant;
  UNTIL PSurMod = NIL;
  PSurIm:=PSurIm.Suivant;
UNTIL (PSurIm = NIL) OR (PSurIm.Valeur.InBloc<>IndBloc);
IF IndObj=NbObj THEN PtIm:=PSurIm;
END;


{=================================================================}
                  FUNCTION ObjetEtBlocCorrect : BOOLEAN;
{=================================================================}

BEGIN
IF ((IndBlocExa=IndBloc) AND (IndObjExa=IndObj)) OR
   ((IndBlocExa=0)       AND (IndObjExa=IndOBJ)) OR
   ((IndBlocExa=IndBloc) AND (IndObjExa=0))      OR
   ((IndBlocExa=0)       AND (IndObjExa=0))      THEN
  ObjetEtBlocCorrect:=TRUE
ELSE ObjetEtBlocCorrect:=FALSE;
END;


{=================================================================}
 PROCEDURE Classification(Choix : INTEGER; Seuil : REAL;
                          VAR NbForme : INTEGER; VAR NbClasse : INTEGER);
{=================================================================}

TYPE
```

```
  DistForme = PACKED RECORD
      I1,I2 : 1..600;
      Dist  : REAL;
      END;
  RecDistForme = RECORD
      Valeur  : DistForme;
      Suivant : RecDistForme;
      END;
  PtIndice = Indice;
  Indice = PACKED RECORD;
      Ind     : 1..600;
      Suivant : Indice;
      END;

VAR
  PtDisFor1,PtDisFor2      : RecDistForme;
  PtDebDisFor,PtPredDisFor : RecDistForme;
  Ind1,Ind2                : 1..600;
  Distance                 : REAL;
  CB                       : PACKED ARRAY[1..600] OF BOOLEAN;
  C                        : ARRAY[1..600] OF PtIndice;
  Pt0,Pt1,Pt2              : Indice;
  Dis                      : ARRAY[1..3] OF 0..127;
  IndDist                  : 1..3;

{=====================}PROCEDURE CalculDistance;{=====================}

BEGIN
FOR Ind1:=1 TO NbForme DO BEGIN NEW(C[Ind1]); C[Ind1]:=NIL; END;
NEW(PtDebDisFor);
WITH PtDebDisFor,Valeur DO
  BEGIN I1:=0; I2:=0; Dist:=-1; Suivant:=NIL; END;
FOR Ind1:= 1 TO NbForme-1 DO
  FOR Ind2:=Ind1+1 TO NbForme DO
    BEGIN
    CASE Choix OF
      1: Distance:=ProdScal(TabAxV[Ind1].VecAxe,TabAxV[Ind2].VecAxe);
      2: Distance:=ProdScal(TabOri[Ind1].VecOri,TabOri[Ind2].VecOri);
      3: BEGIN Distance:=0;
         Dis[1]:=ABS(TabCen[Ind1].Cen.x-TabCen[Ind2].Cen.x);
         Dis[2]:=ABS(TabCen[Ind1].Cen.y-TabCen[Ind2].Cen.y);
         Dis[3]:=ABS(TabCen[Ind1].Cen.z-TabCen[Ind2].Cen.z);
         FOR IndDist:=1 TO 3 DO
           IF Dis[IndDist]>Distance THEN Distance:=Dis[IndDist];
         Distance:=1-(Distance/127);
```

```
        END;
      END;
    IF Distance>=Seuil THEN
      BEGIN
      PtDisFor1:=PtDebDisFor; NEW(PtDisFor2);
      WITH PtDisFor2,Valeur DO
        BEGIN I1:=Ind1; I2:=Ind2; Dist:=Distance;
        IF C[Ind1]= NIL THEN
          BEGIN NEW(Pt0);C[Ind1]:=Pt0;Pt0.Ind:=Ind1;Pt0.Suivant:=NIL;END;
        IF C[INd2]= NIL THEN
          BEGIN NEW(Pt0);C[Ind2]:=Pt0;Pt0.Ind:=Ind2;Pt0.Suivant:=NIL;END;
        IF PtDisFor1.Valeur.Dist<=Distance THEN
          BEGIN Suivant:=PtDebDisFor; PtDebDisFor:=PtDisFor2; END
        ELSE
          BEGIN
          WHILE PtDisFor1.Suivant.Valeur.Dist>Distance DO
            PtDisFor1:=PtDisFor1.Suivant;
          Suivant:=PtDisFor1.Suivant; PtDisFor1.Suivant:=PtDisFor2;
          END;
        END;
      END;
    END;
END;


{============================}PROCEDURE ClasseForme;{========================}

BEGIN PtDisFor1:=PtDebDisFor;
IF PtDisFor1.Valeur.Dist<>-1 THEN
REPEAT
  WITH PtDisFor1,Valeur DO
    BEGIN FOR Ind1:=1 TO NbForme DO CB[Ind1]:=TRUE;
    IF I1<I2 THEN BEGIN Ind1:=I1;Ind2:=I2;END ELSE BEGIN Ind1:=I2;Ind2:=I1;END;
    Pt0:=C[Ind1]; WHILE Pt0.Suivant<>NIL DO Pt0:=Pt0.Suivant;
    Pt0.Suivant:=C[Ind2]; C[Ind2]:=NIL;
    PtDisFor2:=PtDisFor1.Suivant;PtPredDisFor:=PtDisFor1;
    IF PtDisFor2.Valeur.Dist<>-1 THEN
      REPEAT
        WITH PtDisFor2,Valeur DO
          BEGIN
          IF (I1=Ind1)OR(I1=Ind2) THEN
            BEGIN
            IF CB[I2] THEN
              BEGIN PtPredDisFor.Suivant:=PtDisFor2.Suivant;
              DISPOSE(PtDisFor2); CB[I2]:=FALSE; PtDisFor2:=PtPredDisFor;
```

```
            END
          ELSE IF I1=Ind2 THEN I1:=Ind1;
          END
        ELSE
          IF (I2=Ind1)OR(I2=Ind2) THEN
            BEGIN
            IF CB[I1] THEN
              BEGIN PtPredDisFor.Suivant:=PtDisFor2.Suivant;
              DISPOSE(PtDisFor2); CB[I1]:=FALSE; PtDisFor2:=PtPredDisFor;
              END
            ELSE IF I2=Ind2 THEN I2:=Ind1;
            END;
        END;
      PtPredDisFor:=PtDisFor2;PtDisFor2:=PtDisFor2.Suivant;
    UNTIL PtDisFor2.Valeur.Dist=-1;
  END;
 DISPOSE(PtDisFor1); PtDisFor1:=PtDisFor1.Suivant;
UNTIL PtDisFor1.Valeur.Dist=-1;
DISPOSE(PtDisFor1);
END;


{====================}PROCEDURE ConstructionClasse;{====================}

BEGIN NbClasse:=0;
FOR Ind1:=1 TO NbForme DO
  CASE Choix OF
    1: TabAxV[Ind1].IndAxeV:=0;
    2: TabOri[Ind1].IndVOri:=0;
    3: TabCen[Ind1].IndTr:=0;
    END;
FOR Ind1:=1 TO NbForme DO
  BEGIN
  IF C[Ind1]<>NIL THEN
    BEGIN Pt0:=C[Ind1];
    IF Pt0.Suivant<>NIL THEN
      IF Pt0.Suivant.Suivant<>NIL THEN
        BEGIN NbClasse:=NbClasse+1;
        REPEAT
          CASE Choix OF
            1: TabAxV[Pt0.Ind].IndAxeV:=NbClasse;
            2: WITH TabOri[Pt0.Ind],TabAxV[IndAV],TabPSC[IndAV],
                 PSMod.Valeur,PSIm.Valeur DO
                 BEGIN IndVOri:=NbOriVu; ProdVectNor(N1,N2,Vec);
                 PointDroiteInter(P1,P2,D1,D2,N1,N2,Poi);
```

```
                END;
            3: TabCen[Pt0.Ind].IndTr:=NbClasse;
            END;
          DISPOSE(Pt0); Pt0:=Pt0.Suivant;
        UNTIL Pt0=NIL;
        END
      ELSE DISPOSE(Pt0.Suivant)
    ELSE DISPOSE(Pt0);
    END;
  DISPOSE(C[Ind1]);
  END;
END;

{======================== PROCEDURE PRINCIPALE ========================}

BEGIN
CalculDistance;
ClasseForme;
ConstructionClasse;
END;


{======================================================================}
                        PROCEDURE CalculAxeDeVue;
{======================================================================}

BEGIN
FOR IndPairCor:=1 TO NbPairCor DO
  BEGIN
  PSurMod:=TabPSC[IndPairCor].PSMod; PSurIm:=TabPSC[IndPairCor].PSIm;
  WITH PSurMod.Valeur,PSurIm.Valeur,TabAxV[IndPairCor] DO
    BEGIN N1:=Ni1; N2:=Ni2;
    AxeDeVue(Nm1,Nm2,N1,N2,VecAxe);
    END;
  END;
END;


{======================================================================}
                      PROCEDURE CalculOrientationVue;
{======================================================================}

BEGIN NbVecOri:=0;
FOR IndPairCor:=1 TO NbPairCor DO
```

```
  BEGIN
  WITH TabAxV[IndPairCor],TabPSC[IndPairCor].PSMod.Valeur DO
    IF IndAxeV=IndAxV THEN
      BEGIN NbVecOri:=NbVecOri+1; TabOri[NbVecOri].IndAV:=IndPairCor;
      OrientationVue(Nm1,Nm2,N1,N2,TabOri[NbVecOri].VecOri);
      END;
   END;
END;


{=============================================================================}
                          PROCEDURE CalculTranslation;
{=============================================================================}

VAR
  I1,I2,I3,I4 : INTEGER;
  Pt          : POINT;

BEGIN NbCen:=0;
FOR I1:=1 TO NbVecOri-1 DO
  IF TabOri[I1].IndVOri=IndOriVu THEN
    FOR I2:=I1+1 TO NbVecOri DO
      IF TabOri[I2].IndVOri=IndOriVu THEN
        BEGIN I3:=TabOri[I1].IndAV; I4:=TabOri[I2].IndAV;
        IF (TabPSC[I3].PSMod.Valeur.Im1<>TabPSC[I4].PSMod.Valeur.Im1)OR
           (TabPSC[I3].PSMod.Valeur.Im2<>TabPSC[I4].PSMod.Valeur.Im2)THEN
          BEGIN
          WITH TabOri[I1] DO
            MilieuCoupleDroite(10,60,Poi,TabOri[I2].Poi,Vec,TabOri[I2].Vec,Pt);
          IF Pt.z<>-100 THEN
            BEGIN NbCen:=NbCen+1;
            WITH TabCen[NbCen] DO BEGIN Cen:=Pt; IndR1:=I1; IndR2:=I2; END;
            END;
          END;
        END;
END;


{=============================================================================}
                        PROCEDURE CoefficientsTransformation;
{=============================================================================}

VAR
  AxV,VOri            : Vecteur;
  Mil                 : Point;
```

```
  NbMil                : INTEGER;
  C1,C2,C12,CoefT      : INTEGER;
  I1,I2,I3,I4          : INTEGER;
  CosBeta              : REAL;
  SurfCon1,SurfCon2    : BOOLEAN;

BEGIN NbMil:=0; CoefT:=0;
WITH Mil DO BEGIN x:=0; y:=0; z:=0; END;
WITH VOri DO BEGIN u:=0; v:=0; w:=0; END; AxV:=VOri;
FOR I1:=1 TO Obj.NbSur DO
  FOR I2:=0 TO 20 DO TabSC[I1,I2]:=0;
FOR I1:=1 TO NbCen DO
  IF TabCen[I1].IndTr=IndTransf THEN
    BEGIN
    I2:=TabCen[I1].IndR1; I3:=TabOri[I2].IndAV; C1:=TabPSC[I3].CoefSurf;
    WITH TabAxV[I3],VecAxe DO
      BEGIN AxV.u:=AxV.u+u*C1; AxV.v:=AxV.v+v*C1; AxV.w:=AxV.w+w*C1; END;
    WITH TabOri[I2],VecOri DO
      BEGIN VOri.u:=VOri.u+u*C1; VOri.v:=VOri.v+v*C1; VOri.w:=VOri.w+w*C1; END;
    WITH TabPSC[I3].PSMod.Valeur,TabPSC[I3].PSIm.Valeur DO
      BEGIN SurfCon1:=FALSE; SurfCon2:=FALSE;
      FOR I4:=1 TO TabSC[Im1,0] DO IF TabSC[Im1,I4]=Ii1 THEN SurfCon1:=TRUE;
      IF NOT SurfCon1 THEN
        BEGIN TabSC[Im1,0]:=TabSC[Im1,0]+1; TabSC[Im1,TabSc[Im1,0]]:=Ii1; END;
      FOR I4:=1 TO TabSC[Im2,0] DO IF TabSC[Im2,I4]=Ii2 THEN SurfCon2:=TRUE;
      IF NOT SurfCon2 THEN
        BEGIN TabSC[Im2,0]:=TabSC[Im2,0]+1; TabSC[Im2,TabSc[Im2,0]]:=Ii2; END;
      END;
    I2:=TabCen[I1].IndR2; I3:=TabOri[I2].IndAV; C2:=TabPSC[I3].CoefSurf;
    WITH TabAxV[I3],VecAxe DO
      BEGIN AxV.u:=AxV.u+u*C1; AxV.v:=AxV.v+v*C1; AxV.w:=AxV.w+w*C1; END;
    WITH TabOri[I2],VecOri DO
      BEGIN VOri.u:=VOri.u+u*C2; VOri.v:=VOri.v+v*C2; VOri.w:=VOri.w+w*C2; END;
    WITH TabPSC[I3].PSMod.Valeur,TabPSC[I3].PSIm.Valeur DO
      BEGIN SurfCon1:=FALSE; SurfCon2:=FALSE;
      FOR I4:=1 TO TabSC[Im1,0] DO IF TabSC[Im1,I4]=Ii1 THEN SurfCon1:=TRUE;
      IF NOT SurfCon1 THEN
        BEGIN TabSC[Im1,0]:=TabSC[Im1,0]+1; TabSC[Im1,TabSc[Im1,0]]:=Ii1; END;
      FOR I4:=1 TO TabSC[Im2,0] DO IF TabSC[Im2,I4]=Ii2 THEN SurfCon2:=TRUE;
      IF NOT SurfCon2 THEN
        BEGIN TabSC[Im2,0]:=TabSC[Im2,0]+1; TabSC[Im2,TabSc[Im2,0]]:=Ii2; END;
      END;
    C12:=C1+C2; CoefT:=CoefT+C12; NbMil:=NbMil+1;
    WITH TabCen[I1].Cen DO
      BEGIN Mil.x:=Mil.x+x*C12; Mil.y:=Mil.y+y*C12; Mil.z:=Mil.z+z*C12; END;
```

```
    END;
WITH Mil DO BEGIN x:=ROUND(x/CoefT); y:=ROUND(y/CoefT); z:=ROUND(z/CoefT); END;
WITH VOri DO BEGIN u:=u; v:=v; w:=w; NormerVec(VOri); END;
WITH AxV  DO BEGIN u:=u; v:=v; w:=w; NormerVec(AxV); END;
WITH AxV,Rotat DO
  BEGIN CosBeta:=(1-(u*u))**(1/2);
  IF CosBeta<0.01 THEN BEGIN a:=0; b:=-90; g:=0; END
  ELSE
    BEGIN a:=ArcCos(w/CosBeta); b:=ArcCos(CosBeta); g:=0;
    IF v<0 THEN a:=-a; IF u>=0 THEN b:=-b;
    END;
  RotationVect(Rotat,VOri); VOri.w:=0; NormerVec(VOri);
  g:=ArcCos(VOri.u); IF VOri.v>=0 THEN g:=-g;
  END;
WITH Transl,Mil DO BEGIN u:=x; v:=y; w:=z; END; NbCentre:=NbMil;
END;


{=========================================================================}
              FUNCTION CentrePossible(IndSur : INTEGER) : BOOLEAN;
{=========================================================================}

VAR
  Tab                 : Plan;
  TraOri              : Vecteur;
  Centre              : BOOLEAN;

BEGIN
WITH Obj,Sur[IndSur] DO
  IF(Cen.x<0)OR(Cen.x>127)OR(Cen.Y<0)OR(Cen.Y>127)THEN
    BEGIN Centre:=FALSE;
    FOR Yp:=0 TO 127 DO Tab[0,Yp]:=0; FOR Xp:=1 TO 127 DO Tab[Xp]:=Tab[0];
    WITH TraOri DO BEGIN u:=64-Cen.x; v:=64-Cen.y; w:=0; END;
    FOR IndSomLie:= 1 TO NbSomLie DO
      BEGIN P1:=Som[Somlie[IndSomlie]];
      IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
      ELSE P2:=Som[Somlie[IndSomlie+1]];
      Translation(TraOri,P1);Translation(TraOri,P2);ProjecSegment(P1,P2,-1,Tab);
      END;
    NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
    Pi0.x:=64; Pi0.y:=64; Tab[64,64]:=1;
    WHILE Pi0.Suivant<>NIL DO
      WITH Pi0 DO
        BEGIN
        FOR Ind:=1 TO 4 DO
```

```
          BEGIN
          CASE Ind OF
            1: BEGIN Xp:=x+1; Yp:=y; END;
            2: BEGIN Xp:=x-1; Yp:=y; END;
            3: BEGIN Xp:=x; Yp:=y+1; END;
            4: BEGIN Xp:=x; Yp:=y-1; END;
            END;
          IF (NOT Centre)AND(Tab[Xp,Yp]=0)THEN
            BEGIN Pil.x:=Xp; Pil.y:=Yp; NEW(Pi2); Pil.Suivant:=Pi2;
            Pi2.Suivant:=NIL; Pil:=Pi2; Tab[Xp,Yp]:=1;
            Xp:=Xp-64+Cen.x; Yp:=Yp-64+Cen.y;
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              WITH Cen DO
                BEGIN Centre:=TRUE; x:=Xp; y:=Yp; z:=TabDist[Xp,
                                                    Yp]DIV 100; END;
            END;
          END;
        Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
        END;
    DISPOSE(Pi0); CentrePossible:=Centre;
    END
  ELSE CentrePossible:=TRUE;
END;


{=================================================================}
  PROCEDURE DifHautBlocObjet(Choix : INTEGER; Tra : Vecteur; Rot : Rotation;
                                                    VAR Obj : Objet);
{=================================================================}

VAR
  TransfCorrect     : BOOLEAN;
  Vec               : Vecteur;

BEGIN
FOR Yp:=0 TO 127 DO TabHtOb[0,Yp]:=0;
FOR Xp:=1 TO 127 DO TabHtOb[Xp]:=TabHtOb[0];
WITH Obj DO
  BEGIN TransfCorrect:=TRUE;
  FOR IndSom:=1 TO NbSom DO
    BEGIN
    WITH Som[IndSom] DO
      BEGIN RotationPoint(Rot,Som[IndSom]); Translation(Tra,Som[IndSom]);
      x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);
      END;
```

```
    IF Som[IndSom].z<90/CoefZ THEN TransfCorrect:=FALSE;
    END;
  IF TransfCorrect THEN
    FOR IndSur :=1 TO NbSur DO
      WITH Sur[IndSur],Nor DO
        BEGIN RotationVect(Rot,Nor);
        u:=u*CoefX; v:=v*CoefY; w:=w*CoefZ; NormerVec(Nor);
        IF w > 0.2 THEN
          BEGIN
          FOR IndSomLie:= 1 TO NbSomLie DO
            BEGIN P1:=Som[Somlie[IndSomlie]];
            IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
            ELSE P2:=Som[Somlie[IndSomlie+1]];
            ProjecSegment(P1,P2,-1,TabHtOb);
            END;
          RotationPoint(Rot,Cen); Translation(Tra,Cen);
          WITH Cen DO
            BEGIN x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);END;
          IF (CentrePossible(IndSur)) THEN
          IF (TabHtOb[Cen.x,Cen.y]=0) THEN
            BEGIN NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
            WITH Cen DO
              BEGIN Pi0.x:=x; Pi0.y:=Y;
              CASE Choix OF
                1 : BEGIN
                    NbPi0b:=NbPi0b+1; TabHtOb[x,y]:=z*100;
                    DifHt:=DifHt+ABS(TabDist[x,y]-TabHtOb[x,y]);
                    IF ABS(TabBloc[x,y])=IndBloc THEN NbPi0bBc:=NbPi0bBc+1;
                    END;
                2 : BEGIN TabDist[x,y]:=0; TabHtOb[x,y]:=1; END;
                3 : BEGIN TabHtOb[x,y]:=z*100;
                    IF TabDist[x,y]<>0 THEN
                      BEGIN NbPi0b:=NbPi0b+1;
                      DifHt:=DifHt+ABS(TabDist[x,y]-TabHtOb[x,y]);
                      END;
                    END;
              END;
              END;
            WHILE Pi0.Suivant<>NIL DO
              WITH Pi0 DO
                BEGIN
                FOR Ind:=1 TO 4 DO
                  BEGIN
                  CASE Ind OF
                    1: BEGIN Xp:=x+1; Yp:=y; END;
```

```
                    2: BEGIN Xp:=x-1; Yp:=y; END;
                    3: BEGIN Xp:=x; Yp:=y+1; END;
                    4: BEGIN Xp:=x; Yp:=y-1; END;
                    END;
                  IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
                    IF (TabHtOb[Xp,Yp]<=0) THEN
                      BEGIN
                      IF TabHtOb[Xp,Yp]=0 THEN
                        BEGIN Pi1.x:=Xp;Pi1.y:=Yp;NEW(Pi2);Pi1.Suivant:=Pi2;
                        Pi2.Suivant:=NIL; Pi1:=Pi2;
                        END;
                      CASE Choix OF
                        1: BEGIN NbPiOb:=NbPiOb+1;
                           WITH Vec,Cen DO BEGIN u:=(x-Xp);v:=(y-Yp);w:=z; END;
                           TabHtOb[Xp,Yp]:=ROUND((ProdScal(Vec,Nor)*100)/Nor.w);
                           IF TabHtOb[Xp,Yp]<=0 THEN TabHtOb[Xp,Yp]:=1;
                           DifHt:=DifHt+ABS(TabDist[Xp,Yp]-TabHtOb[Xp,Yp]);
                           IF TabBloc[Xp,Yp]=IndBloc THEN NbPiObBc:=NbPiObBc+1;
                           END;
                        2: BEGIN
                           IF (TabBloc[Xp,Yp]=IndBloc)AND
                              (TabDist[Xp,Yp]<>0) THEN SurfBloc:=SurfBloc-1;
                           TabDist[Xp,Yp]:=0; TabHtOb[Xp,Yp]:=1;
                           END;
                        3: BEGIN
                           WITH Vec,Cen DO BEGIN u:=(x-Xp);v:=(y-Yp);w:=z; END;
                           TabHtOb[Xp,Yp]:=ROUND((ProdScal(Vec,Nor)*100)/Nor.w);
                           IF TabHtOb[Xp,Yp]<=0 THEN TabHtOb[Xp,Yp]:=1;
                           IF TabDist[Xp,Yp]<>0 THEN
                             BEGIN NbPiOb:=NbPiOb+1;
                             DifHt:=DifHt+ABS(TabDist[Xp,Yp]-TabHtOb[Xp,Yp]);
                             END;
                           END;
                        END;
                      END;
                  Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
                  END;
              DISPOSE(Pi0);
              END
            END;
          END;
      END;
  END;
END;
```

```
{=============================================================================}
          PROCEDURE DifHautSurfaceObjet(TabSuCo : TabSurfConnecte);
{=============================================================================}

BEGIN NbPiSuOb:=0; ComSurf:=[];
WITH Obj DO
FOR IndSur:=1 TO NbSur DO
  IF (Sur[IndSur].Nor.w>0.2)AND(TabSuCo[IndSur,0]<>0) THEN
    FOR Ind:=1 TO TabSuCo[IndSur,0] DO
      BEGIN ComSurf:=ComSurf+[TabSuCo[IndSur,Ind]];
      WITH Ptsur[TabSuCo[IndSur,Ind]] DO
        BEGIN NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
        Pi0.x:=x; Pi0.y:=y;
        IF TabDist[x,y]<>0 THEN
          BEGIN NbPiSuOb:=NbPiSuOb+1;
          DifHt:=DifHt+ABS(TabDist[x,y]-TabHtOb[x,y]);
          END;
        TabHtOb[x,y]:=-1;
        END;
      WHILE Pi0.Suivant<>NIL DO
        WITH Pi0 DO
          BEGIN
          FOR Ind1:=1 TO 4 DO
            BEGIN
            CASE Ind1 OF
              1: BEGIN Xp:=x+1; Yp:=y; END;
              2: BEGIN Xp:=x-1; Yp:=y; END;
              3: BEGIN Xp:=x; Yp:=y+1; END;
              4: BEGIN Xp:=x; Yp:=y-1; END;
              END;
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              IF(TabHtOb[Xp,Yp]>=0)AND(TabSurfP[Xp,Yp]=TabSuCo[IndSur,Ind])THEN
                BEGIN Pi1.x:=Xp;Pi1.y:=Yp;NEW(Pi2);Pi1.Suivant:=Pi2;
                Pi2.Suivant:=NIL; Pi1:=Pi2;
                IF TabDist[Xp,Yp]<>0 THEN
                  BEGIN NbPiSuOb:=NbPiSuOb+1;
                  DifHt:=DifHt+ABS(TabDist[Xp,Yp]-TabHtOb[Xp,Yp]);
                  END;
                TabHtOb[Xp,Yp]:=-1;
                END;
            END;
          Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
          END;
      DISPOSE(Pi0);
      END;
```

```
END;

{=============================================================================}
 PROCEDURE DifferenceHauteur(Tra : Vecteur; Rot : Rotation; VAR Obj : Objet);
{=============================================================================}

BEGIN RESET(FicObjet); READ(FicObjet,Obj);
DifHt:=0; NbPiOb:=0; NbPiObBc:=0; NbPiSuOb:=0;
DifHautBlocObjet(1,Tra,Rot,Obj);
IF NbPiOb>0 THEN
  IF NbPiObBc/NbPiOb>0.80 THEN
    BEGIN Extension(3,TabHtOb); DifHautSurfaceObjet(TabSC); END;
END;

{=============================================================================}
                        PROCEDURE SelectionTransformationModele;
{=============================================================================}

VAR
  CombSurfExiste : BOOLEAN;

BEGIN CombSurfExiste:=FALSE; IndComSurf:=0;
IF NbComSurf[IndBloc,IndObj]>0 THEN
  REPEAT IndComSurf:=IndComSurf+1;
    IF ComSurf*TComSurf[IndBloc,IndObj,IndComSurf]<>[] THEN
      BEGIN CombSurfExiste:=TRUE;
      TComSurf[IndBloc,IndObj,IndComSurf]:=ComSurf+
                                          TComSurf[IndBloc,IndObj,IndComSurf];
      END;
  UNTIL (CombSurfExiste)OR(IndComSurf=NbComSurf[IndBloc,IndObj]);
IF (NbPiOb>0)AND(NbPiSuOb>0) THEN
  IF NbPiObBc/NbPiOb>0.80 THEN
    BEGIN
    IF NOT CombSurfExiste THEN
      BEGIN NbComSurf[IndBloc,IndObj]:=NbComSurf[IndBloc,IndObj]+1;
      IndComSurf:=NbComSurf[IndBloc,IndObj];
      TComSurf[IndBloc,IndObj,IndComSurf]:=ComSurf;
      END;
    WITH TabCObSu[IndBloc,IndObj,IndComSurf] DO
      BEGIN DifHt:=ROUND(DifHt/(NbPiOb+NbPiSuOb));
      IF (NbCentre>NbCen*1.25)OR((NbCentre>NbCen*0.75)AND(DifHt<DifH)) THEN
        BEGIN IndOb:=IndObj; Rot:=Rotat; Tra:=Transl; NbCen:=NbCentre; IndCl:=0;
        DifH:=DifHt; SurOb:=NbPiOb; SurObBc:=NbPiObBc; TabSuCo:=TabSC;
```

```
      END;
      END;
    END;
END;

{=========================================================================}
                FUNCTION NombreSolutionsPossibles : INTEGER;
{=========================================================================}

VAR
  NbSolPos : INTEGER;

BEGIN NbSolPos:=0;
FOR IndObj:=1 TO NbObj DO
FOR IndComSurf:=1 TO NbComSurf[IndBloc,IndObj] DO
  WITH TabCObSu[IndBloc,IndObj,IndComSurf] DO
    IF IndCl=0 THEN NbSolPos:=NbSolPos+1;
NombreSolutionsPossibles:=NbSolPos;
END;

{=========================================================================}
                    PROCEDURE SuppressionMeilleurSolution;
{=========================================================================}

BEGIN DifHt:=1000000; IndObjet:=0; IndClasse:=IndClasse+1; NbCentre:=0;
FOR IndObj:=1 TO NbObj DO
FOR IndComSurf:=1 TO NbComSurf[IndBloc,IndObj] DO
  WITH TabCObSu[IndBloc,IndObj,IndComSurf] DO
    IF IndCl=0 THEN
      IF (DifH<DifHt*0.95)OR((DifH<DifHt*1.05)AND(NbCen>NbCentre)) THEN
        BEGIN DifHt:=DifH; IndObjet:=IndOb; IndComSurf1:=IndComSurf;
        NbCentre:=NbCen;
        END;
IF IndObjet<>0 THEN
  BEGIN CLOSE(FicObjet);
  IF IndObjet>9 THEN Num:=CHR((IndObjet DIV 10)+48)+CHR((IndObjet MOD 10)+48)
  ELSE Num:=CHR(IndObjet+48);
  NomFicObj:=NomObj+Num+'.CAD';
  OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
  WITH TabCObSu[IndBloc,IndObjet,IndComSurf1] DO
    BEGIN IndCl:=IndClasse;
    DifHautBlocObjet(2,Tra,Rot,Obj);
    END;
```

```
  END;
END;

{=============================================================================}
                    PROCEDURE ActualisationAutresSolutions;
{=============================================================================}

BEGIN
FOR IndObj:=1 TO NbObj DO
FOR IndComSurf:=1 TO NbComSurf[IndBloc,IndObj] DO
  WITH TabCObSu[IndBloc,IndObj,IndComSurf] DO
    IF IndCl=0 THEN
      BEGIN CLOSE(FicObjet);
      IF IndOb>9 THEN Num:=CHR((IndOb DIV 10)+48)+CHR((IndOb MOD 10)+48)
      ELSE Num:=CHR(IndOb+48);
      NomFicObj:=NomObj+Num+'.CAD';
      OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
      DifHt:=0; NbPiOb:=0; NbPiSuOb:=0;
      DifHautBlocObjet(3,Tra,Rot,Obj);
      IF NbPiOb/SurOb<0.20 THEN IndCl:=-1
      ELSE
        BEGIN Extension(3,TabHtOb); DifHautSurfaceObjet(TabSuCo);
        IF NbPiSuOb/SurOb<0.20 THEN IndCl:=-1
        ELSE DifH:=ROUND(DifHt/(NbPiOb+NbPiSuOb));
        END;
      END;
END;

{=============================================================================}
                              PROCEDURE Resultats;
{=============================================================================}

VAR
  Temps,Heures,Minutes,Secondes,Centiemes : INTEGER;

BEGIN
FOR IndBloc:=2 TO NbBloc DO
  BEGIN
  WRITELN(FicText,'     ******************** ',IndBloc:3,' Bloc *************'
                 ,'**********');
  WRITELN;
  WRITELN(FicText,'-------------------------------------------------------------'
                 ,'------');
```

```
  WRITELN(FicText,'|  No   |  Position  |  Rotation  | Translation|Indice qu'
                 ,'alite|');
  WRITELN(FicText,'| Modele| Empilement | Rx  Ry  Rz | Tx  Ty  Tz |  (10E-5 '
                 ,'M)   |');
  WRITELN(FicText,'-------------------------------------------------------'
                 ,'------');
  FOR IndObj:=1 TO NbObj DO
  FOR IndComSurf:=1 TO NbComSurf[IndBloc,IndObj] DO
    WITH TabCObSu[IndBloc,IndObj,IndComSurf] DO
      IF IndCl>-1 THEN
        BEGIN WRITE(FicText,'|  ',IndOb:3,'  |    ',IndCl:3,'      |');
        WITH Rot DO WRITE(FicText,a:4,b:4,g:4,'|');
        WITH Tra DO WRITE(FicText,ROUND(u):4,ROUND(v):4,ROUND(W-100):4);
        WRITELN(FicText,'|    ',DifH:6,'    |');
        END;
  WRITELN(FicText,'-------------------------------------------------------'
                 ,'------');
  WRITELN(FicText);WRITELN(FicText);
  END;
Temps:=CLOCK-HeureDebut;
Heures:=Temps DIV 3600000;
Minutes:=(Temps DIV 60000)MOD 60;
Secondes:=(Temps DIV 1000)MOD 60;
Centiemes:=(Temps MOD 1000)DIV 10;
WRITE(FicText,'TEMPS CPU ',Temps:10,' = ');
IF Heures=0 THEN WRITE(FicText,'00:')
ELSE
  BEGIN
  IF Heures<10 THEN WRITE(FicText,'0',Heures:1,':')
  ELSE WRITE(FicText,Heures:2,':');
  END;
IF Minutes=0 THEN WRITE(FicText,'00:')
ELSE
  BEGIN
  IF Minutes<10 THEN WRITE(FicText,'0',Minutes:1,':')
  ELSE WRITE(FicText,Minutes:2,':');
  END;
IF Secondes=0 THEN WRITE(FicText,'00.')
ELSE
  BEGIN
  IF Secondes<10 THEN WRITE(FicText,'0',Secondes:1,'.')
  ELSE WRITE(FicText,Secondes:2,'.');
  END;
IF Centiemes=0 THEN WRITELN(FicText,'00')
ELSE
```

```
  BEGIN
  IF Centiemes<10 THEN WRITELN(FicText,'0',Centiemes:1)
  ELSE WRITELN(FicText,Centiemes:2);
  END;
END;


{*****************************************************************************}
                              PROGRAMME PRINCIPAL
{*****************************************************************************}

BEGIN
Initialisation;
LectureSceneEtudiee;
FOR IndBloc:=NbBloc DOWNTO 2 DO
  BEGIN
  FOR IndObj:= 1 TO NbObj DO
    BEGIN
    LectureModeleObjet;
    MiseCorrespondanceSurfaces;
    IF (NbPairCor>3) AND (ObjetEtBlocCorrect) THEN
      BEGIN
      CalculAxeDeVue;
      Classification(1,Seuil2,NbPairCor,NbAxV);
      FOR IndAxV:=1 TO NbAxV DO
        BEGIN
        CalculOrientationVue;
        Classification(2,Seuil3,NbVecOri,NbOriVu);
        FOR IndOriVu:=1 TO NbOriVu DO
          BEGIN
          CalculTranslation;
          Classification(3,Seuil4,NbCen,NbTransf);
          FOR IndTransf:=1 TO NbTransf DO
            BEGIN
            CoefficientsTransformation;
            DifferenceHauteur(Transl,Rotat,Obj);
            SelectionTransformationModele;
            END;
          END;
        END;
      END;
    END;
  WHILE (SurfBloc/TabPiBc[IndBloc]>0.2)AND(NombreSolutionsPossibles>0) DO
    BEGIN
    SuppressionMeilleurSolution;
```

```
    ActualisationAutresSolutions;
    END;
  END;
Resultats;
END.
```

H. "IMPRIMEIMAGE" PROGRAM

This program is used to create a visual representation of many images created during the process. The result of this program is a text file. Three different representations can be created depending on the nature of images:

1. a representation of the range data by associating a grey level at each pixel of the image (see Fig. A-8a)

2. a representation of the normals at each pixel of a range data image (see Fig. A-8b)

3. a representation corresponding to the writing of the values at each pixel of the image (see Fig. A-8c). To have a clear result, the program asks the value of the pixels belonging to the background. At the place of these pixels just a point will be printed. Furthermore, the pixels having a value equal to 1 will be represented by a black square. This last particularity is used to represent the polyhedra.

Note: this program can superpose many representations (see Fig. A-8d).

Parameters

- Name of the text file.

- Names of the different images to be processed.

The other parameters depend of the representation of the image:

1. representation of the range data
   - No special parameters

2. representation of the normal at each pixel of the image
   - Millimetre distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis), and between two levels of the range data (Z-axis).

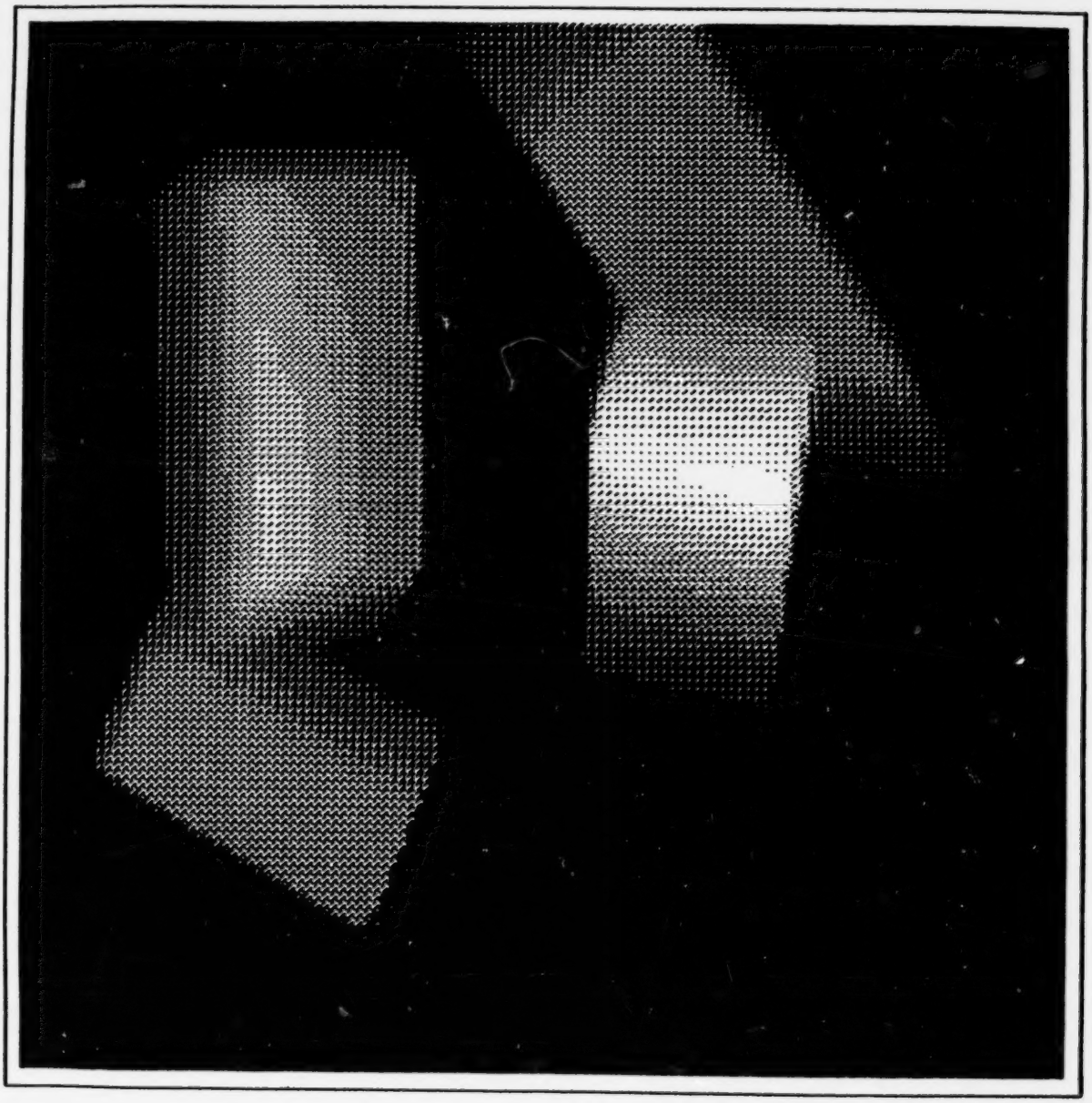
3. representation of the value in each pixel of the image
   - Value of the background

Input Files

- Files containing the different images to be processed

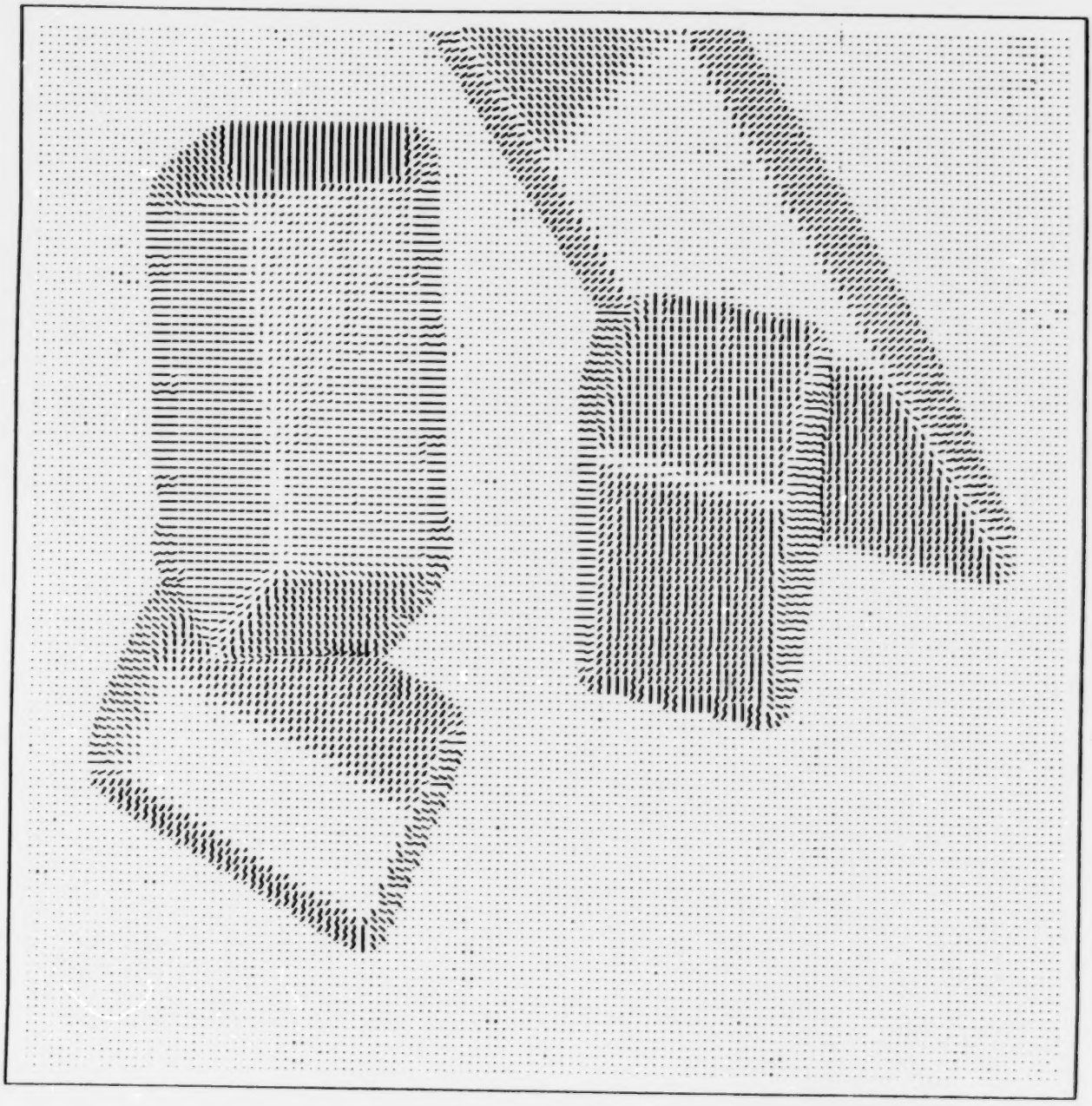
Output File

- Text file corresponding to the visual representation of the studied images.



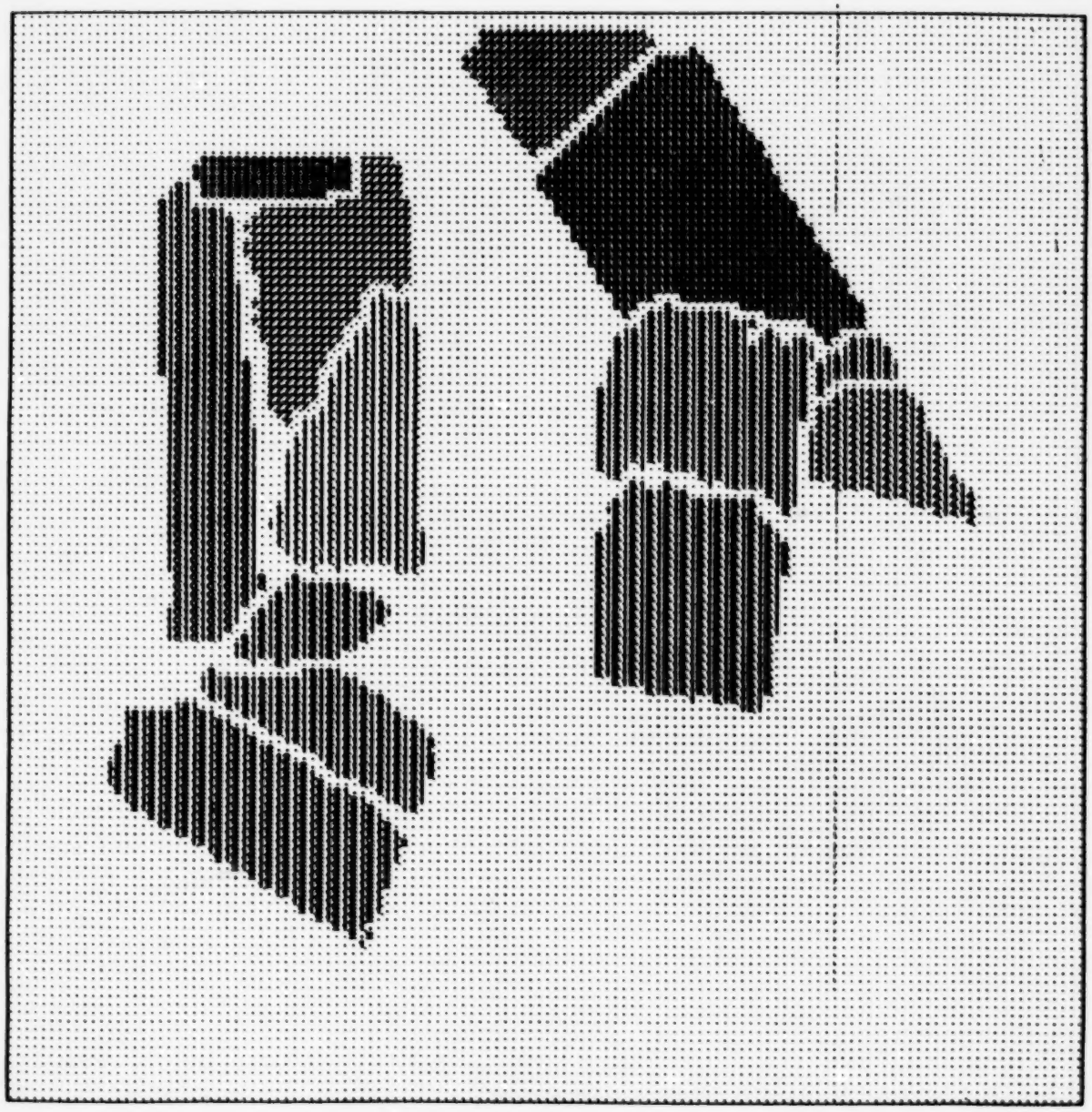
(a)

Figure A-8. Output files of the "IMPRIMEIMAGE" program. (a) Range data. (b) Normal at each pixels of the scene. (c) Value at each pixel of the scene. (d) Superposition of two images (Figs. A-8c and A-9).



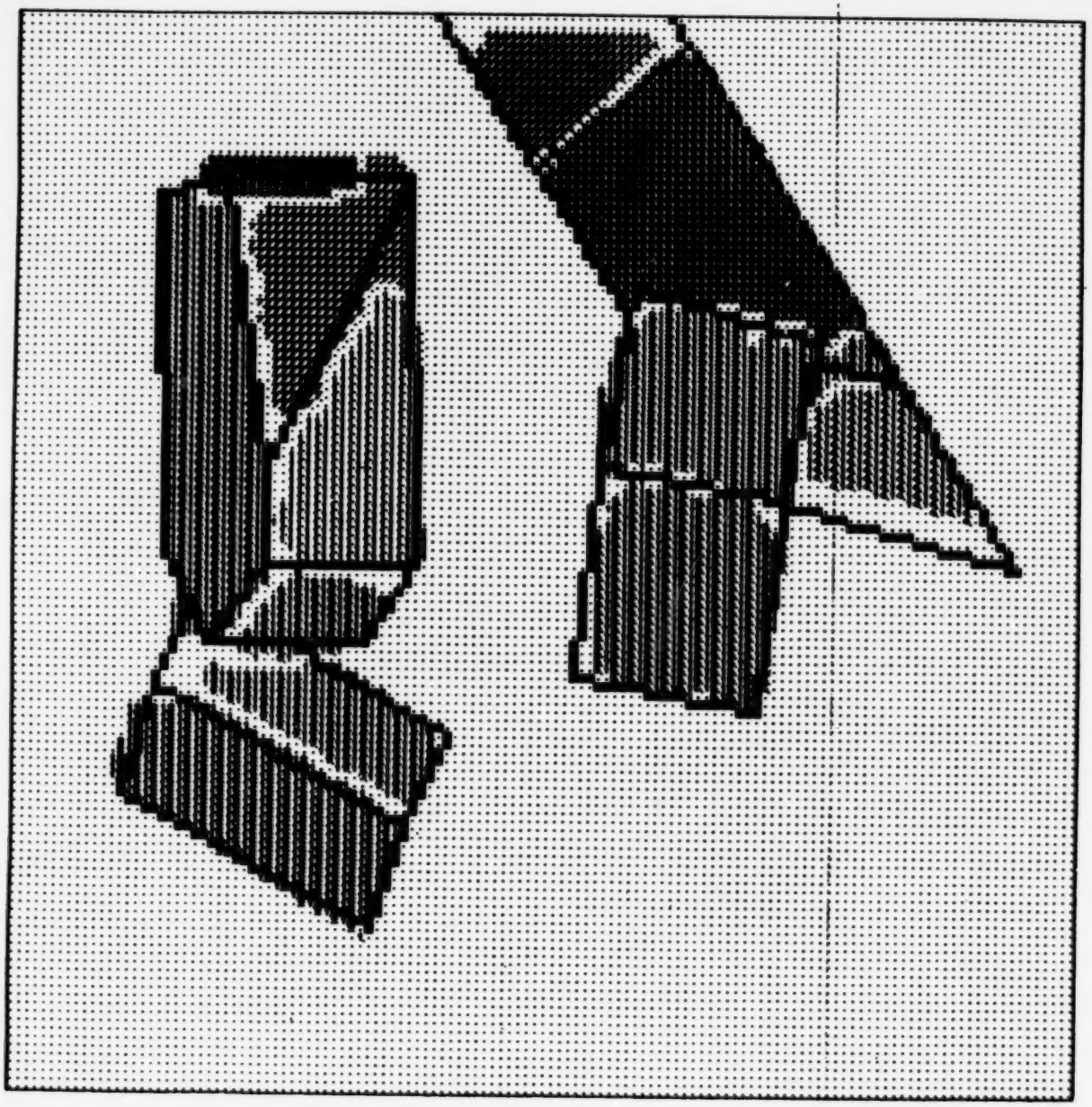
(b)

Figure A-8. (Continued).



(c)

Figure A-8. (Continued).



(d)

Figure A-8. (Concluded).

```
{*****************************************************************************}
                    PROGRAM ImpressionImage(INPUT,OUTPUT);
{*****************************************************************************}


{========================= DECLARATION DE TYPES ============================}

TYPE
  Degre = -180..180;
  Point = RECORD
        x,y,z : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w : REAL;
        END;
  Rotation = RECORD
        a,b,g : INTEGER;
        END;
  CONV = RECORD CASE BOOLEAN OF
        TRUE :(TEX:ARRAY[1..107] OF CHAR);
        FALSE :(BOO:PACKED ARRAY[1..856] OF BOOLEAN);
        END;
  Bytes = -32766..32765;
  Ligne = PACKED ARRAY[0..127] OF Bytes;
  Plan  = ARRAY[0..127] OF Ligne;
  FichierPlan = FILE OF Ligne;

{======================= DECLARATION DE VARIABLES ==========================}

VAR
  NomficIm,Tampon,NomFicText : PACKED ARRAY[1..12] OF CHAR;
  FicImage                   : FichierPlan;
  FicText                    : TEXT;
  A                          : PACKED ARRAY[1..768,1..642] OF BOOLEAN;
  B                          : CONV;
  CoefX,CoefY,CoefZ          : REAL;
  Choix,Charac               : CHAR;
  TabDist                    : Plan;
  P1,P2                      : Point;
  I                          : INTEGER;
  Sol                        : INTEGER;

{====================== DECLARATION DE PROCEDURES ==========================}

PROCEDURE LectureImage( VAR Fic : FichierPlan; Tab : Plan);EXTERN;
```

```
FUNCTION MTH$RANDOM(VAR K:INTEGER):REAL;                                EXTERN;

{=========================================================================}
                          PROCEDURE InitTabBool;
{=========================================================================}

VAR
  I,J : INTEGER;

BEGIN
FOR J:=1 TO 642 DO A[1,J]:=FALSE;
FOR J:=2 TO 768 DO A[J]:=A[1];
FOR I:=1 TO 768 DO BEGIN A[I,1]:=TRUE; A[I,640]:=TRUE; END;
FOR J:=1 TO 640 DO BEGIN A[1,J]:=TRUE; A[768,J]:=TRUE; END;
END;

{=========================================================================}
                          PROCEDURE EcritTabBool;
{=========================================================================}

VAR
  I,J,K,L,M,Val,LongVal,Chiffre : INTEGER;

BEGIN
FOR I:=0 TO 127 DO
 FOR J:=0 TO 127 DO
  BEGIN Val:=TabDist[I,J]; IF Val=Sol THEN Val:=0; IF Val<0 THEN Val:=-1;
  CASE Val OF
   -1: BEGIN
       FOR K:=1 TO 6 DO BEGIN A[I*6+K,J*5+1]:=TRUE; A[I*6+K,J*5+5]:=TRUE; END;
       FOR L:=1 TO 5 DO BEGIN A[I*6+1,J*5+L]:=TRUE; A[I*6+6,J*5+L]:=TRUE; END;
       END;
    0: A[I*6+4,J*5+3]:=TRUE;
    1: BEGIN
       FOR K:=1 TO 6 DO
         FOR L:=1 TO 5 DO A[I*6+K,J*5+L]:=TRUE;
       END;
   OTHERWISE
       BEGIN
       FOR K:=0 TO 3 DO IF Val>=10**K THEN LongVal:=1+K;
       FOR K:=0 TO LongVal-1 DO
        IF (J+K)MOD(LongVal)=0 THEN
          BEGIN
          Chiffre:=(Val-((Val DIV 10**(K+1))*10**(K+1)))DIV 10**K;
          CASE Chiffre OF
```

```
0 : BEGIN
    A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+1]:=TRUE;
    A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+1]:=TRUE;A[I*6+3,J*5+4]:=TRUE;
    A[I*6+4,J*5+1]:=TRUE;A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+2]:=TRUE;
    A[I*6+5,J*5+3]:=TRUE;
    END;
1 : BEGIN
    A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+3]:=TRUE;A[I*6+2,J*5+2]:=TRUE;
    A[I*6+3,J*5+3]:=TRUE;A[I*6+4,J*5+3]:=TRUE;A[I*6+5,J*5+3]:=TRUE;
    END;
2 : BEGIN
    A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+1]:=TRUE;
    A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+3]:=TRUE;A[I*6+4,J*5+2]:=TRUE;
    A[I*6+5,J*5+1]:=TRUE;A[I*6+5,J*5+2]:=TRUE;A[I*6+5,J*5+3]:=TRUE;
    A[I*6+5,J*5+4]:=TRUE;
    END;
3 : BEGIN
    A[I*6+1,J*5+1]:=TRUE;A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;
    A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+2]:=TRUE;A[I*6+3,J*5+3]:=TRUE;
    A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+1]:=TRUE;A[I*6+5,J*5+2]:=TRUE;
    A[I*6+5,J*5+3]:=TRUE;
    END;
4 : BEGIN
    A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+2]:=TRUE;A[I*6+2,J*5+3]:=TRUE;
    A[I*6+3,J*5+1]:=TRUE;A[I*6+3,J*5+3]:=TRUE;A[I*6+4,J*5+1]:=TRUE;
    A[I*6+4,J*5+2]:=TRUE;A[I*6+4,J*5+3]:=TRUE;A[I*6+4,J*5+4]:=TRUE;
    A[I*6+5,J*5+3]:=TRUE;
    END;
5 : BEGIN
    A[I*6+1,J*5+1]:=TRUE;A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;
    A[I*6+2,J*5+1]:=TRUE;A[I*6+3,J*5+1]:=TRUE;A[I*6+3,J*5+2]:=TRUE;
    A[I*6+3,J*5+3]:=TRUE;A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+1]:=TRUE;
    A[I*6+5,J*5+2]:=TRUE;A[I*6+5,J*5+3]:=TRUE;
    END;
6 : BEGIN
    A[I*6+1,J*5+2]:=TRUE;A[I*6+2,J*5+1]:=TRUE;A[I*6+3,J*5+1]:=TRUE;
    A[I*6+3,J*5+2]:=TRUE;A[I*6+3,J*5+3]:=TRUE;A[I*6+4,J*5+1]:=TRUE;
    A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+2]:=TRUE;A[I*6+5,J*5+3]:=TRUE;
    END;
7 : BEGIN
    A[I*6+1,J*5+1]:=TRUE;A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;
    A[I*6+1,J*5+4]:=TRUE;A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+3]:=TRUE;
    A[I*6+4,J*5+2]:=TRUE;A[I*6+5,J*5+2]:=TRUE;
    END;
8 : BEGIN
```

```
                A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+1]:=TRUE;
                A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+2]:=TRUE;A[I*6+3,J*5+3]:=TRUE;
                A[I*6+4,J*5+1]:=TRUE;A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+2]:=TRUE;
                A[I*6+5,J*5+3]:=TRUE;
                END;
            9 : BEGIN
                A[I*6+1,J*5+2]:=TRUE;A[I*6+1,J*5+3]:=TRUE;A[I*6+2,J*5+1]:=TRUE;
                A[I*6+2,J*5+4]:=TRUE;A[I*6+3,J*5+2]:=TRUE;A[I*6+3,J*5+3]:=TRUE;
                A[I*6+3,J*5+4]:=TRUE;A[I*6+4,J*5+4]:=TRUE;A[I*6+5,J*5+2]:=TRUE;
                A[I*6+5,J*5+3]:=TRUE;
                END;
              END;
          IF K=0 THEN A[I*6+6,J*5+5]:=TRUE;
          END;
       END;
     END;
  END;
END;

{=========================================================================}
        FUNCTION  LissageMed(x,y : INTEGER; VAR Tab : Plan):INTEGER;
{=========================================================================}

VAR
  Valeur,NbCaseExa : INTEGER;
  V                : ARRAY[1..10] OF INTEGER;
  Decale           : BOOLEAN;
  Ind1,Ind2,I,J    : INTEGER;

BEGIN V[1]:=-9999999;
FOR I:=-1 TO 1 DO
  FOR J:=-1 TO 1 DO
    BEGIN Valeur:=Tab[x+I,y+J]; NbCaseExa:=(1+I)*3+2+J;
    Ind1:=1;Decale:=FALSE;
    REPEAT
      IF V[Ind1]<Valeur THEN
        BEGIN FOR Ind2:= NbCaseExa+1 DOWNTO Ind1+1 DO V[Ind2]:=V[Ind2-1];
        V[Ind1]:=Valeur;Decale:=TRUE;
        END;
      Ind1:=Ind1+1;
    UNTIL (Ind1>5)OR(Decale);
    END;
LissageMed:=V[5];
END;
```

```
{=============================================================================}
                          PROCEDURE CalculHauteur;
{=============================================================================}

VAR
  I,J,K,L,M,Ip,Jp,Ii,Ji,NbPt : INTEGER;
  Ht,HtMin,HtMax             : INTEGER;
  R                          : REAL;

BEGIN HtMin:=99999999; HtMax:=-99999999;
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    BEGIN Ht:=LissageMed(I,J,TabDist);
    IF Ht<HtMin THEN HtMin:=Ht; IF Ht>HtMax THEN HtMax:=Ht;
    END;
FOR I:=1 TO 126 DO
  BEGIN
  FOR J:=1 TO 126 DO
    BEGIN NbPt:=0; Ip:=I*6; Jp:=J*5; K:=3;
    Ht:=LissageMed(I,J,TabDist);
    Ht:=ROUND(((Ht-HtMin)*30)/(HtMax-HtMin));
    IF Ht<=15 THEN
      BEGIN
      FOR L:=0 TO 5 DO
        FOR M:= 0 TO 4 DO A[Ip+L,Jp+M]:=TRUE;
      WHILE NbPt<Ht DO
        BEGIN
        R:=MTH$RANDOM(K); Ji:=ABS(K MOD 5);
        R:=MTH$RANDOM(K); Ii:=ABS((K+Ji) MOD 6);
        IF A[Ip+Ii,Jp+Ji] THEN
          BEGIN A[Ip+Ii,Jp+Ji]:=FALSE; NbPt:=NbPt+1; END;
        END
      END
    ELSE
      BEGIN
      WHILE NbPt<30-Ht DO
        BEGIN
        R:=MTH$RANDOM(K); Ji:=ABS(K MOD 5); Ji:=4-Ji;
        R:=MTH$RANDOM(K); Ii:=ABS((K+Ji) MOD 6); Ii:=5-Ii;
        IF NOT A[Ip+Ii,Jp+Ji] THEN
          BEGIN A[Ip+Ii,Jp+Ji]:=TRUE; NbPt:=NbPt+1; END;
        END;
      END;
    END;
  END;
```

```
END;

{=============================================================================}
                  PROCEDURE ProjecArete(P1,P2:Point);
{=============================================================================}

VAR
  Xi,Xs,Yi,Ys : INTEGER;
  I,J : INTEGER;
  Dx,Dy,Pente,Constante : REAL;

BEGIN
WITH P1 DO
  BEGIN Dx:=x-P2.x; Dy:=y-P2.y;
  IF (Dx=0) AND (Dy=0) THEN A[x,y]:=TRUE
  ELSE
    IF ABS(Dx)>ABS(Dy) THEN
      BEGIN
      IF x<P2.x THEN BEGIN Xi:=x;Xs:=P2.x; END ELSE BEGIN Xi:=P2.x;Xs:=x; END;
      Pente:=Dy/Dx; Constante:=y-Pente*x;
      FOR I:=Xi TO Xs DO
        BEGIN J:=ROUND(Pente*I+Constante);
        IF (J>=1)AND(J<=640)AND(I>=1)AND(I<=768) THEN A[I,J]:=TRUE;
        END;
      END
    ELSE
      BEGIN
      IF y<P2.y THEN BEGIN Yi:=y;Ys:=P2.y; END ELSE BEGIN Yi:=P2.y;Ys:=y; END;
      Pente:=Dx/Dy; Constante:=x-Pente*y;
      FOR J:=Yi TO Ys DO
        BEGIN I:=ROUND(Pente*J+Constante);
        IF (I>=1)AND(I<=768)AND(J>=1)AND(J<=640) THEN A[I,J]:=TRUE;
        END;
      END;
  END;
END;

{=============================================================================}
                          PROCEDURE CalculNormale;
{=============================================================================}

VAR
  I,J,K,L     : INTEGER;
```

```
  HGrad,VGrad : INTEGER;
  Alpha,Beta  : REAL;
  Nor         : Vecteur;
  Ht          : ARRAY[-1..1,-1..1] OF INTEGER;

BEGIN
FOR I:=1 TO 126 DO
  FOR J:=1 TO 126 DO
    BEGIN HGrad:=0; VGrad:=0;
    FOR K:=-1 TO 1 DO
      FOR L:=-1 TO 1 DO Ht[K,L]:=TabDist[I+K,J+L];
    HGrad:=HGrad+Ht[-1,-1]+Ht[0,-1]+Ht[1,-1]-Ht[-1,1]-Ht[0,1]-Ht[1,1];
    VGrad:=VGrad-Ht[1,-1]-Ht[1,0]-Ht[1,1]+Ht[-1,-1]+Ht[-1,0]+Ht[-1,1];
    IF VGrad=0 THEN
      BEGIN Alpha:=3.1415926534/2; IF HGrad<0 THEN Alpha:=-Alpha; END
    ELSE
      BEGIN Alpha:=ARCTAN((HGrad*CoefX)/(VGrad*CoefY));
      IF VGrad<0 THEN Alpha:=3.1415926534+Alpha;
      END;
    IF (HGrad=0)AND(VGrad=0) THEN Beta:=3.1415926534/2
    ELSE Beta:=ARCTAN((6*100)/(SQRT((VGrad/CoefX)**2+(HGrad/CoefY)**2)*CoefZ));
    WITH Nor DO
      BEGIN u:=Cos(Alpha)*Cos(Beta);v:=Sin(Alpha)*Cos(Beta);w:=Sin(Beta); END;
    WITH P1 DO BEGIN x:=I*6+4; y:=J*5+3; z:=0; END;
    WITH P2,Nor DO BEGIN x:=I*6+4+ROUND(u*5); y:=J*5+3+ROUND(v*4); z:=0; END;
    ProjecArete(P1,P2);
    END;
END;


{=============================================================================}
                              PROCEDURE ImpTabBool;
{=============================================================================}

VAR
  I,J,K : INTEGER;

BEGIN
WRITELN(FicText); WRITELN(FicText);
FOR I:=1 TO 768 DO
  BEGIN
  FOR J:=1 TO 107 DO
    FOR K:=1 TO 8 DO
      CASE K OF
        7 : B.BOO[(J-1)*8+7]:=TRUE;
```

```
        8 : B.BOO[(J-1)*8+8]:=FALSE;
        OTHERWISE B.BOO[(J-1)*8+K]:=A[I,(J-1)*6+K];
      END;
  WRITE(FicText,CHR(5));
  FOR J:=1 TO 107 DO
  WRITE(FicText,B.TEX[J]);
  WRITELN(FicText);
  END;
END;


{*****************************************************************************}
{                          PROGRAMME PRINCIPAL                                }
{*****************************************************************************}


BEGIN WRITELN('**** VISUALISATION DES NORMALES ****');
WRITELN;WRITELN;WRITELN;
WRITE(' Nom du fichier text ? ');READLN(NomFicText);
OPEN(FicText,NomFicText,NEW,270); REWRITE(FicText);
FOR I:=1 TO 63 DO WRITELN(FicText);
REPEAT
  InitTabBool;
  REPEAT
    WRITELN;WRITELN;
    WRITE(' Nom du fichier Image ? ');READLN(NomFicIm);
    OPEN(FicImage,NomFicIm,OLD);LectureImage(FicImage,TabDist);CLOSE(FicImage);
    WRITE(' Imprime V)aleur, H)auteur, N)ormale ? ');READLN(Charac);
    CASE Charac OF
     'V','v': BEGIN WRITE(' Indice du sol ? ');READLN(Sol);
              EcritTabBool;
              END;
     'H','h': CalculHauteur;
     'N','n': BEGIN
              WRITE(' Valeur Coefficients en X,Y,Z ? ');
              READLN(CoefX,CoefY,CoefZ);
              CalculNormale;
              END;
     END;
    WRITELN;
    WRITE(' Autre fichier a superposer ? ');READLN(Choix);
  UNTIL (Choix='n')OR(Choix='N');
  ImpTabBool;
  WRITE(' Autre fichier ? ');READLN(Choix);
UNTIL (Choix='n')OR(Choix='N');
END.
```

I. "PROJECTIONOBJET" PROGRAM

This program allows to create an image representing the result of the analysis of a scene (see Fig. A-9). For this, the models detected are rotated, translated and projected in the XY-Plan. This program does not draw the hidden parts of the objects, for
this reason the different models must be entered in the order corresponding to the overlap of the scene objects from top to bottom. This order is given by the result of the "ANALYSE" program.

Parameters

- Name of the created image.
- Millimetre distances between two pixels along the vertical image axis (X-axis), the horizontal image axis (Y-axis) and between two levels of the range data (Z-axis).
- Name of the CAD-Model of objects detected in the scene.
- Values of the translations to be applied to the CAD-Models.
- Values of the rotations to be applied to the CAD-Models.

Input Files

- Files containing the CAD-Model of detected scene objects.

Output File

- File containing the image representing the analysis result.

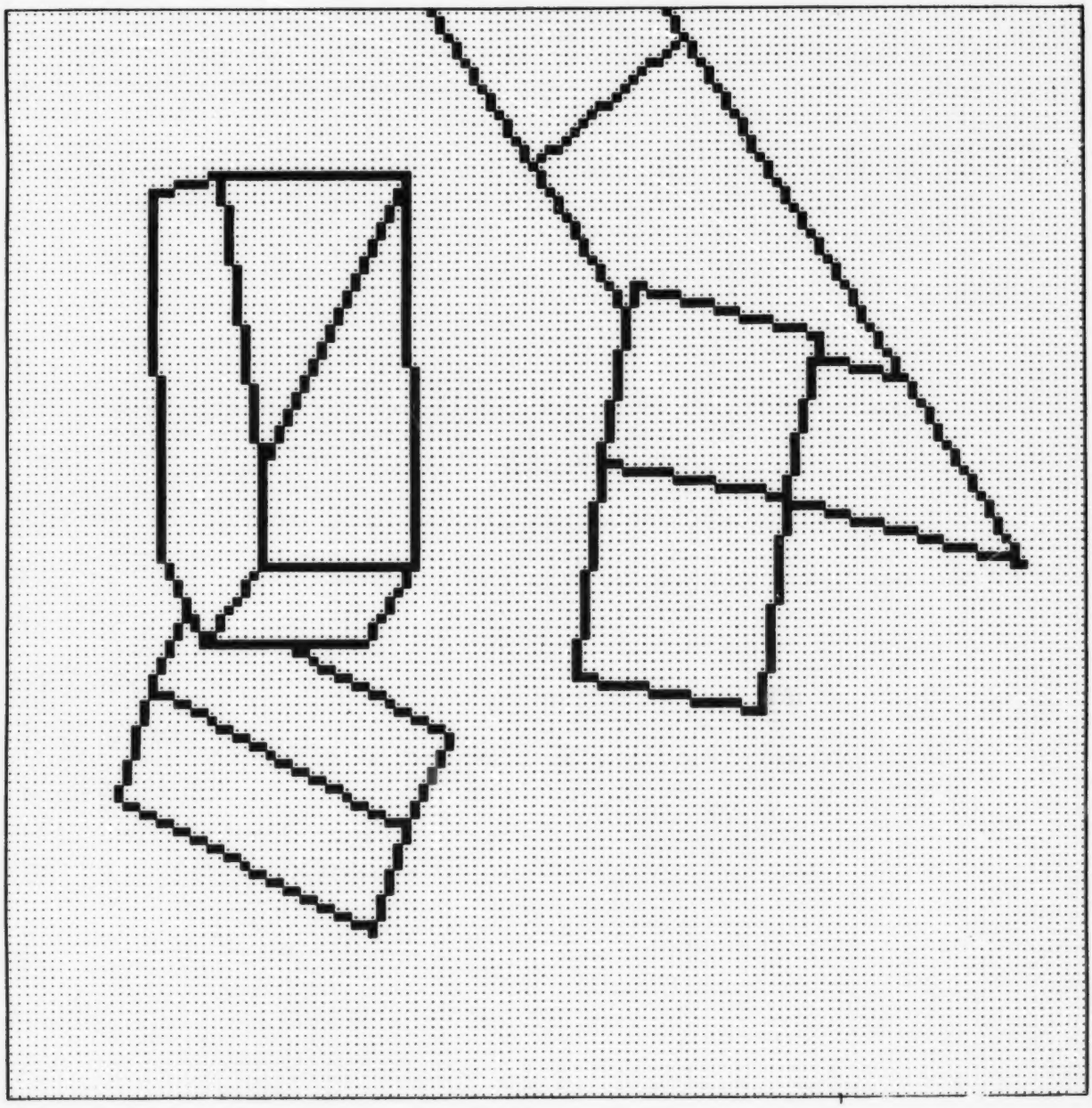


Figure A-9. Visualisation of the result of the analysis of the scene corresponding to the Fig. A-6.

```
{*****************************************************************************}
                    PROGRAM ProjectionObjet(INPUT,OUTPUT);
{*****************************************************************************}


{========================= DECLARATION DE TYPES ============================}

TYPE
  Degre = -180..180;
  Point = RECORD
        x,y,z     : INTEGER;
        END;
  Vecteur = RECORD
        u,v,w     : REAL;
        END;
  Rotation = RECORD
        a,b,g     : INTEGER;
        END;
  ElementSurface = RECORD
        x,y       : INTEGER;
        Suivant   : ElementSurface;
        END;
  Surface = RECORD
        Nor       : Vecteur;
        Cen       : Point;
        Taille    : INTEGER;
        NbSomLie  : 1..20;
        SomLie    : ARRAY[1..20] OF 0..100;
        NbSurAdj  : 1..20;
        SurAdj    : ARRAY[1..20] OF 0..20;
        END;
  Objet = RECORD
        NbSom     : 1..100;
        Som       : ARRAY[1..100] OF Point;
        NbSur     : 1..20;
        Sur       : ARRAY[1..20] OF Surface;
        END;
  Bytes       = -32768..32767;
  Ligne       = PACKED ARRAY[0..127] OF Bytes;
  Plan        = ARRAY[0..127] OF Ligne;
  FichierPlan = FILE OF Ligne;


{======================= DECLARATION DE VARIABLES ==========================}

VAR
```

```
  Obj                                  : Objet;
  IndSur,IndSurAdj,IndSurAdj2          : 1..10;
  IndSom,IndSomLie                     : 1..100;
  NomficObj,Tampon,NomFicProj          : VARYING[12] OF CHAR;
  FicProj                              : FichierPlan;
  FicObjet                             : FILE OF Objet;
  Tra                                  : Vecteur;
  Rot                                  : Rotation;
  P1,P2                                : Point;
  TabXY,TabSurf                        : Plan;
  CoefX,CoefY,CoefZ                    : REAL;
  Choix                                : CHAR;
  Li                                   : Ligne;
  I,J,Xp,Yp,Ind                        : INTEGER;
  Pi0,Pi1,Pi2,Debut                    : ElementSurface;
  Alpha,Beta                           : REAL;


{====================== DECLARATION DE PROCEDURES ======================}

PROCEDURE RotationVect(Rot : Rotation; VAR Vec : Vecteur);              EXTERN;
PROCEDURE RotationPoint(Rot : Rotation; VAR P : Point);                EXTERN;
PROCEDURE Translation(Ttr : Vecteur; VAR P : Point);                   EXTERN;
PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER; VAR Tab:PLAN);     EXTERN;
PROCEDURE EcritureImage(VAR Fic : FichierPlan; Tab : Plan);            EXTERN;
PROCEDURE NormerVec(VAR V : Vecteur);                                  EXTERN;


{=============================================================================}
              FUNCTION CentrePossible(IndSur : INTEGER) : BOOLEAN;
{=============================================================================}

VAR
  Tab                 : Plan;
  TraOri              : Vecteur;
  Centre              : BOOLEAN;

BEGIN
WITH Obj,Sur[IndSur] DO
  IF(Cen.x<0)OR(Cen.x>127)OR(Cen.Y<0)OR(Cen.Y>127)THEN
    BEGIN Centre:=FALSE;
    FOR Yp:=0 TO 127 DO Tab[0,Yp]:=0; FOR Xp:=1 TO 127 DO Tab[Xp]:=Tab[0];
    WITH TraOri DO BEGIN u:=64-Cen.x; v:=64-Cen.y; w:=0; END;
    FOR IndSomLie:= 1 TO NbSomLie DO
      BEGIN P1:=Som[Somlie[IndSomlie]];
```

```
      IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
      ELSE P2:=Som[Somlie[IndSomlie+1]];
      Translation(TraOri,P1);Translation(TraOri,P2);ProjecSegment(P1,P2,-1,Tab);

      END;
    NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
    Pi0.x:=64; Pi0.y:=64; Tab[64,64]:=1;
    WHILE Pi0.Suivant<>NIL DO
      WITH Pi0 DO
        BEGIN
        FOR Ind:=1 TO 4 DO
          BEGIN
          CASE Ind OF
            1: BEGIN Xp:=x+1; Yp:=y; END;
            2: BEGIN Xp:=x-1; Yp:=y; END;
            3: BEGIN Xp:=x; Yp:=y+1; END;
            4: BEGIN Xp:=x; Yp:=y-1; END;
            END;
          IF (NOT Centre)AND(Tab[Xp,Yp]=0)THEN
            BEGIN Pi1.x:=Xp; Pi1.y:=Yp; NEW(Pi2); Pi1.Suivant:=Pi2;
            Pi2.Suivant:=NIL; Pi1:=Pi2; Tab[Xp,Yp]:=1;
            Xp:=Xp-64+Cen.x; Yp:=Yp-64+Cen.y;
            IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
              WITH Cen DO BEGIN Centre:=TRUE; x:=Xp; y:=Yp; END;
            END;
          END;
        Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
        END;
    DISPOSE(Pi0); CentrePossible:=Centre;
    END
  ELSE CentrePossible:=TRUE;
END;


{=============================================================================}
     PROCEDURE ProjObj(Tra : Vecteur; Rot : Rotation; VAR Obj : Objet);
{=============================================================================}

BEGIN
WITH Obj DO
  BEGIN
  FOR IndSom:=1 TO NbSom DO
    WITH Som[IndSom] DO
      BEGIN RotationPoint(Rot,Som[IndSom]); Translation(Tra,Som[IndSom]);
      x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);
```

```
    END;
  FOR IndSur :=1 TO NbSur DO
    WITH Sur[IndSur],Nor DO
      BEGIN RotationVect(Rot,Nor);
      u:=u*CoefX; v:=v*CoefY; w:=w*CoefZ; NormerVec(Nor);
      IF w > 0.1 THEN
        BEGIN
        FOR J:=0 TO 127 DO TabSurf[0,j]:=0;
        FOR I:=1 TO 127 DO TabSurf[I]:=TabSurf[0];
        FOR IndSomLie:= 1 TO NbSomLie DO
          BEGIN P1:=Som[Somlie[IndSomlie]];
          IF IndSomLie = NbSomlie THEN P2:=Som[Somlie[1]]
          ELSE P2:=Som[Somlie[IndSomlie+1]];
          ProjecSegment(P1,P2,1,TabXY); ProjecSegment(P1,P2,1,TabSurf);
          END;
        RotationPoint(Rot,Cen); Translation(Tra,Cen);
        WITH Cen DO
          BEGIN x:=ROUND(x/CoefX); y:=ROUND(y/CoefY); z:=ROUND(z/CoefZ);END;
        IF (CentrePossible(IndSur)) THEN
          IF (TabSurf[Cen.x,Cen.y]=0) THEN
            BEGIN NEW(Pi0); NEW(Pi1); Pi0.Suivant:=Pi1; Pi1.Suivant:=NIL;
            WITH Cen DO
              BEGIN Pi0.x:=x; Pi0.y:=Y; TabSurf[x,y]:=1; TabXY[x,y]:=-1; END;
            WHILE Pi0.Suivant<>NIL DO
              WITH Pi0 DO
                BEGIN
                FOR Ind:=1 TO 4 DO
                  BEGIN
                  CASE Ind OF
                    1: BEGIN Xp:=x+1; Yp:=y; END;
                    2: BEGIN Xp:=x-1; Yp:=y; END;
                    3: BEGIN Xp:=x; Yp:=y+1; END;
                    4: BEGIN Xp:=x; Yp:=y-1; END;
                    END;
                  IF(Xp>0)AND(Xp<127)AND(Yp>0)AND(Yp<127)THEN
                    IF (TabSurf[Xp,Yp]=0) THEN
                      BEGIN Pi1.x:=Xp;Pi1.y:=Yp;NEW(Pi2);Pi1.Suivant:=Pi2;
                      Pi2.Suivant:=NIL; Pi1:=Pi2; TabSurf[Xp,Yp]:=1;
                      IF TabXY[Xp,Yp]=0 THEN TabXY[Xp,Yp]:=-1;
                      END;
                  END;
                Debut:=Suivant;DISPOSE(Pi0);Pi0:=Debut;
                END;
            DISPOSE(Pi0);
            END
```

```
        END;
      END;
  END;
END;


{*****************************************************************************}
                              PROGRAMME PRINCIPAL
{*****************************************************************************}

BEGIN WRITELN('**** PROJECTION D''UN OBJET ****');
WRITELN;WRITELN;WRITELN;
WRITE(' Nom du fichier projection ? ');READLN(NomFicProj);
WRITE(' Valeur Coefficients en X,Y,Z ? ');READLN(CoefX,CoefY,CoefZ);
REPEAT
  WRITELN;WRITELN;
  WRITE(' Nom du fichier Objet ? ');READLN(NomFicObj);
  IF LENGTH(NomFicObj)=0 THEN NomFicObj:=Tampon
  ELSE Tampon:=NomFicObj;
  OPEN(FicObjet,NomFicObj,OLD);RESET(FicObjet);READ(FicObjet,Obj);
  CLOSE(FicObjet);
  WRITE(' Parametres de la translation ? ');WITH Tra DO READLN(u,v,w);
  WRITE(' Parametres de la rotation ? ');WITH Rot DO READLN(a,b,g);
  ProjObj(Tra,Rot,Obj);
  WRITE(' Autre projection ? ');READLN(CHOIX);
UNTIL (Choix='n')OR(CHOIX='N');
FOR I:=0 TO 127 DO
  FOR J:=0 TO 127 DO IF TabXY[I,J]<0 THEN TabXY[I,J]:=0;
OPEN(FicProj,NomFicProj);REWRITE(FicProj);
EcritureImage(FicProj,TabXY);
END.
```

J. "UTILITAIRES" FILE

This file contains many procedures corresponding to specific computations used in many different programs. For this reason, all the previous programs must be link with the result of the compilation of this file.

```
{*************************************************************************}
                    MODULE Utilitaire (INPUT,OUTPUT);
{*************************************************************************}


{*************************************************************************
 1 : PROCEDURE NormerVec(VAR V : Vecteur);
 2 : FUNCTION  ProdScal(V1,V2 : Vecteur) :REAL;
 3 : FUNCTION  ArcCos(Alpha : REAL) : Degre;
 4 : PROCEDURE SauveProdScal(Scal1,Scal2 : REAL; VAR Vi,Vj : Vecteur);
 5 : PROCEDURE ProdVect(Vi,Vj : Vecteur; VAR Vk : Vecteur);
 6 : PROCEDURE ProdVectNor(Vi,Vj : Vecteur; VAR Vk : Vecteur);
 7 : FUNCTION  Determinant(a,b,c,d,e,f,g,h,i : REAL) : REAL;
 8 : PROCEDURE RotationVect(R : Rotation; VAR V : Vecteur);
 9 : PROCEDURE RotationPoint(R : Rotation; VAR P : Point);
10 : PROCEDURE Translation(T : Vecteur; VAR P : Point);
11 : PROCEDURE AxeDeVue(Vi1,Vi2:Vecteur; VAR Vf1,Vf2,VAxe:Vecteur);
12 : PROCEDURE OrientationVue(Vi1,Vi2:Vecteur; VAR Vf1,Vf2,VAxe:Vecteur);
13 : PROCEDURE PointDroiteInter(P1,P2:Point; D1,D2:REAL; V1,V2:Vecteur;
                                                        VAR P0:Point);
14 : FUNCTION  DistancePlan(Pc,Ps : Point; N : Vecteur) : REAL;
15 : FUNCTION  CarreDistPoint(P1,P2 : Point) : INTEGER;
16 : PROCEDURE MilieuCoupleDroite(Seuil,LargOb:INTEGER; P1,P2:Point;
                                          N1,N2:Vecteur; VAR P0:Point);
17 : PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER; VAR TAB:Plan);
18 : PROCEDURE LectureImage(VAR Fic : FichierPlan;VAR Tab : Plan);
19 : PROCEDURE EcritureImage(VAR Fic : FichierPlan;VAR Tab : Plan);
*************************************************************************}


{======================= DECLARATION DE TYPES ========================}

TYPE
  BYTES = -32768..32767;
  Ligne = PACKED ARRAY[0..127] OF BYTES;
  Plan = ARRAY[0..127] OF Ligne;
  FichierPlan = FILE OF Ligne;
  Degre = -180..180;
  Point = RECORD
          x,y,z : INTEGER;
          END;
  Vecteur = RECORD
          u,v,w : REAL;
          END;
  Rotation = RECORD
```

```
          a,b,g : INTEGER;
          END;

{=========================================================================}
                    PROCEDURE NormerVec(VAR V : Vecteur);
{=========================================================================}

VAR
  Norme : REAL;

BEGIN
WITH V DO
  BEGIN Norme:=SQRT(u*u+v*v+w*w);
  IF Norme<>0 THEN BEGIN u:=u/Norme; v:=v/Norme; w:=w/Norme; END;
  END;
END;


{=========================================================================}
                  FUNCTION ProdScal(V1,V2 : Vecteur) :REAL;
{=========================================================================}

BEGIN
ProdScal:=V1.u*V2.u+V1.v*V2.v+V1.w*V2.w;
END;


{=========================================================================}
                    FUNCTION ArcCos(Alpha : REAL) : Degre;
{=========================================================================}

CONST
  Coef180Pi=57.29577637;

VAR
  ArcCos1 : Degre;

BEGIN
IF Alpha = 0 THEN ArcCos:=90
ELSE
  BEGIN
  ArcCos1:=ROUND(ARCTAN(SQRT(ABS(1-Alpha*Alpha))/Alpha)*Coef180Pi);
  IF (ArcCos1=0) AND (Alpha<0) THEN ArcCos:=180
  ELSE IF ArcCos1 >= 0 THEN ArcCos:=ArcCos1 ELSE ArcCos:=180+ArcCos1;
```

```
  END;
END;


{=========================================================================}
      PROCEDURE SauveProdScal(Scal1,Scal2 : REAL; VAR Vi,Vj : Vecteur);
{=========================================================================}

VAR
  Cons1,Coef1,Coef2 : REAL;
  Vt : Vecteur;
  Ang1,Ang2 : REAL;

BEGIN
Ang1:=ARCTAN(Vi.w/SQRT(Vi.u**2+Vi.v**2));
Ang2:=ARCTAN(Vj.w/SQRT(Vj.u**2+Vj.v**2));
IF ((Ang1>10)AND(Ang2>10))OR((Ang1<10)AND(Ang2<10)) THEN
  BEGIN
  Cons1:=1-Scal2*Scal2;
  Coef1:=SQRT(ABS((1-Scal1*Scal2+SQRT(Cons1*(1-Scal1*Scal1)))/(2*Cons1)));
  Coef2:=SQRT(ABS((1-Scal1*Scal2-SQRT(Cons1*(1-Scal1*Scal1)))/(2*Cons1)));
  IF Scal1<Scal2 THEN Coef2:=-Coef2;
  Vt:=Vi;
  WITH Vi DO
    BEGIN u:=Coef1*u+Coef2*Vj.u; v:=Coef1*v+Coef2*Vj.v; w:=Coef1*w+Coef2*Vj.w;
    END;
  WITH Vj DO
    BEGIN u:=Coef2*Vt.u+Coef1*u; v:=Coef2*Vt.v+Coef1*v; w:=Coef2*Vt.w+Coef1*w;
    END;
  END
ELSE
  BEGIN
  Coef2:=SQRT((1-Scal1*Scal1)/(1-Scal2*Scal2));
  Coef1:=Scal1-Coef2*Scal2;
  IF Vj.w<Vi.w THEN
    WITH Vj DO
      BEGIN u:=Coef1*Vi.u+Coef2*u;v:=Coef1*Vi.v+Coef2*v;w:=Coef1*Vi.w+Coef2*w;
      END
  ELSE
    WITH Vi DO
      BEGIN u:=Coef2*u+Coef1*Vj.u;v:=Coef2*v+Coef1*Vj.v;w:=Coef2*w+Coef1*Vj.w;
      END;
  END;
NormerVec(Vi);NormerVec(Vj);
END;
```

```
{=====================================================================}
          PROCEDURE ProdVect(Vi,Vj : Vecteur; VAR Vk : Vecteur);
{=====================================================================}

BEGIN
WITH Vk DO
  BEGIN
  u:=Vi.v*Vj.w-Vi.w*Vj.v; v:=Vi.w*Vj.u-Vi.u*Vj.w; w:=Vi.u*Vj.v-Vi.v*Vj.u;
  END;
END;


{=====================================================================}
        PROCEDURE ProdVectNor(Vi,Vj : Vecteur; VAR Vk : Vecteur);
{=====================================================================}

BEGIN
WITH Vk DO
  BEGIN
  u:=Vi.v*Vj.w-Vi.w*Vj.v; v:=Vi.w*Vj.u-Vi.u*Vj.w; w:=Vi.u*Vj.v-Vi.v*Vj.u;
  END;
NormerVec(Vk);
END;


{=====================================================================}
          FUNCTION Determinant(a,b,c,d,e,f,g,h,i : REAL) : REAL;
{=====================================================================}

BEGIN
Determinant:=a*(e*i-h*f)-b*(d*i-g*f)+c*(d*h-g*e);
END;


{=====================================================================}
          PROCEDURE RotationVect(R : Rotation; VAR V : Vecteur);
{=====================================================================}

CONST
  CoefPi180 = 0.01745329238;

VAR
  AngRad,Cosinus,Sinus : REAL;
  ut,vt,wt : REAL;
```

```
BEGIN
WITH R,V DO
  BEGIN
  AngRad:=a*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad);
  vt:=v; v:=Cosinus*v-Sinus*w; w:=Sinus*vt+Cosinus*w;
  AngRad:=b*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad);
  wt:=w; w:=Cosinus*w-Sinus*u; u:=Sinus*wt+Cosinus*u;
  AngRad:=g*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad);
  ut:=u; u:=Cosinus*u-Sinus*v; v:=Sinus*ut+Cosinus*v;
  END;
NormerVec(V);
END;


{=================================================================}
          PROCEDURE RotationPoint(R : Rotation; VAR P : Point);
{=================================================================}

CONST
  CoefPi180 = 0.01745329238;

VAR
  AngRad,Cosinus,Sinus : REAL;
  Xi,Yi,Zi,Xt,Yt,Zt : REAL;

BEGIN
WITH R,P DO
  BEGIN Xi:=x; Yi:=y; Zi:=z;
  AngRad:=a*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad); Yt:=Yi;
  Yi:=Cosinus*Yi-Sinus*Zi; Zi:=Sinus*Yt+Cosinus*Zi;
  AngRad:=b*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad); Zt:=Zi;
  Zi:=Cosinus*Zi-Sinus*Xi; Xi:=Sinus*Zt+Cosinus*Xi;
  AngRad:=g*CoefPi180; Cosinus:=Cos(AngRad); Sinus:=Sin(AngRad); Xt:=Xi;
  Xi:=Cosinus*Xi-Sinus*Yi; Yi:=Sinus*Xt+Cosinus*Yi;
  x:=ROUND(Xi); y:=ROUND(Yi); z:=ROUND(Zi);
  END;
END;


{=================================================================}
            PROCEDURE Translation(T : Vecteur; VAR P : Point);
{=================================================================}

BEGIN
WITH T,P DO
```

```
  BEGIN x:=x+ROUND(u); y:=y+ROUND(v); z:=z+ROUND(w); END;
END;

{=========================================================================}
      PROCEDURE AxeDeVue(Vi1,Vi2:Vecteur; VAR Vf1,Vf2,VAxe:Vecteur);
{=========================================================================}

VAR
  Vi3,Vf3 : Vecteur;

BEGIN
SauveProdScal(ProdScal(Vi1,Vi2),ProdScal(Vf1,Vf2),Vf1,Vf2);
ProdVectNor(Vi1,Vi2,Vi3); ProdVectNor(Vf1,Vf2,Vf3);
WITH VAxe DO
  BEGIN
  u:=Determinant(Vf1.w,Vi1.v,Vi1.w,Vf2.w,Vi2.v,Vi2.w,Vf3.w,Vi3.v,Vi3.w);
  v:=Determinant(Vi1.u,Vf1.w,Vi1.w,Vi2.u,Vf2.w,Vi2.w,Vi3.u,Vf3.w,Vi3.w);
  w:=Determinant(Vi1.u,Vi1.v,Vf1.w,Vi2.u,Vi2.v,Vf2.w,Vi3.u,Vi3.v,Vf3.w);
  END;
NormerVec(Vaxe);
END;

{=========================================================================}
   PROCEDURE OrientationVue(Vi1,Vi2:Vecteur; VAR Vf1,Vf2,VAxe:Vecteur);
{=========================================================================}

VAR
  Vi3,Vf3 : Vecteur;

BEGIN
SauveProdScal(ProdScal(Vi1,Vi2),ProdScal(Vf1,Vf2),Vf1,Vf2);
ProdVectNor(Vi1,Vi2,Vi3); ProdVectNor(Vf1,Vf2,Vf3);
WITH VAxe DO
  BEGIN
  u:=Determinant(Vf1.u,Vi1.v,Vi1.w,Vf2.u,Vi2.v,Vi2.w,Vf3.u,Vi3.v,Vi3.w);
  v:=Determinant(Vi1.u,Vf1.u,Vi1.w,Vi2.u,Vf2.u,Vi2.w,Vi3.u,Vf3.u,Vi3.w);
  w:=Determinant(Vi1.u,Vi1.v,Vf1.u,Vi2.u,Vi2.v,Vf2.u,Vi3.u,Vi3.v,Vf3.u);
  END;
NormerVec(Vaxe);
END;
```

```
{=========================================================================}
 PROCEDURE PointDroiteInter(P1,P2:Point; D1,D2:REAL; V1,V2:Vecteur;
                                                        VAR PO:Point);
{=========================================================================}

VAR
  VO : Vecteur;
  Coef1,Coef2,ScalV1V2 : REAL;

BEGIN
WITH VO,P1 DO
  BEGIN
  u:=x+D1*V1.u-P2.x-D2*V2.u;
  v:=y+D1*V1.v-P2.y-D2*V2.v;
  w:=z+D1*V1.w-P2.z-D2*V2.w;
  END;
ScalV1V2:=ProdScal(V1,V2);
Coef1:=ProdScal(V2,VO)/(1-ScalV1V2*ScalV1V2);Coef2:=Coef1*ScalV1V2;
WITH PO,V1 DO
  BEGIN
  x:=ROUND(Coef2*u-Coef1*V2.u+P1.x+D1*u);
  y:=ROUND(Coef2*v-Coef1*V2.v+P1.y+D1*v);
  z:=ROUND(Coef2*w-Coef1*V2.w+P1.z+D1*w);
  END;
END;


{=========================================================================}
        FUNCTION DistancePlan(Pc,Ps : Point; N : Vecteur) : REAL;
{=========================================================================}

VAR
  V : Vecteur;

BEGIN
WITH V,Pc DO BEGIN u:=x-Ps.x; v:=y-Ps.y; w:=z-Ps.z; END;
DistancePlan:=ProdScal(N,V);
END;


{=========================================================================}
 PROCEDURE MilieuCoupleDroite(Seuil,LargOb:INTEGER; P1,P2:Point;
                                              N1,N2:Vecteur; VAR PO:Point);
{=========================================================================}

VAR
```

```
  N0,N3 : Vecteur;
  Scal0,Scal1,Scal2,Scal3 : REAL;
  Denom,Num1,Num2 : REAL;

BEGIN P0.z:=-100; Scal1:=ProdScal(N1,N2);
IF (ArcCos(Scal1)>10)AND(ArcCos(Scal1)<170) THEN
  BEGIN ProdVectNor(N1,N2,N0);
  WITH N3,P1 DO BEGIN u:=P2.x-x; v:=P2.y-y; w:=P2.z-z; END;
  Scal0:=ProdScal(N0,N3);
  IF Scal0<=Seuil THEN
    BEGIN
    Scal2:=ProdScal(N1,N3); Scal3:=ProdScal(N2,N3);
    Denom:=1-(Scal1*Scal1); Num1:=Scal3*Scal1-Scal2;
    IF Num1/Denom<= LargOb THEN
      BEGIN Num2:=Scal3-Scal2*Scal1;
      IF Num2/Denom<= LargOb THEN
        WITH P0,N1 DO
          BEGIN
          x:=ROUND(((P1.x+P2.x)*Denom-Num1*u-Num2*N2.u)/(2*Denom));
          y:=ROUND(((P1.y+P2.y)*Denom-Num1*v-Num2*N2.v)/(2*Denom));
          z:=ROUND(((P1.z+P2.z)*Denom-Num1*w-Num2*N2.w)/(2*Denom));
          END;
      END;
    END;
  END;
END;


{=========================================================================}
            FUNCTION CarreDistPoint(P1,P2 : Point) : INTEGER;
{=========================================================================}

VAR
  Dx,Dy,Dz : INTEGER;

BEGIN
WITH P1 DO
  BEGIN Dx:=x-P2.x; Dy:=y-P2.y; Dz:=z-P2.z; END;
CarreDistPoint:=Dx*Dx+Dy*Dy+Dz*Dz;
END;
```

```
{=============================================================================}
    PROCEDURE ProjecSegment(P1,P2:Point; Poids:INTEGER; VAR TAB:Plan);
{=============================================================================}

VAR
  Xi,Xs,Yi,Ys : INTEGER;
  I,J : INTEGER;
  A,B,Pente,Constante : REAL;

BEGIN
WITH P1 DO
  BEGIN A:=x-P2.x; B:=y-P2.y;
  IF (A=0) AND (B=0) THEN
    BEGIN
    IF (x>=0)AND(x<=127)AND(y>=0)AND(y<=127) THEN
      IF TAB[x,y]=0 THEN TAB[x,y]:=Poids;
    END
  ELSE
    IF ABS(A)>ABS(B) THEN
      BEGIN
      IF x<P2.x THEN BEGIN Xi:=x;Xs:=P2.x; END ELSE BEGIN Xi:=P2.x;Xs:=x; END;
      Pente:=B/A; Constante:=y-Pente*x;
      FOR I:=Xi TO Xs DO
        BEGIN J:=ROUND(Pente*I+Constante);
        IF (J>=0)AND(J<=127)AND(I>=0)AND(I<=127) THEN
          IF TAB[I,J]=0 THEN TAB[I,J]:=Poids;
        END;
      END
    ELSE
      BEGIN
      IF y<P2.y THEN BEGIN Yi:=y;Ys:=P2.y; END ELSE BEGIN Yi:=P2.y;Ys:=y; END;
      Pente:=A/B; Constante:=x-Pente*y;
      FOR J:=Yi TO Ys DO
        BEGIN I:=ROUND(Pente*J+Constante);
        IF (I>=0)AND(I<=127)AND(J>=0)AND(J<=127) THEN
          IF TAB[I,J]=0 THEN TAB[I,J]:=Poids;
        END;
      END;
  END;
END;
```

```
{=============================================================================}
      PROCEDURE LectureImage(VAR Fic : FichierPlan;VAR Tab : Plan);
{=============================================================================}

VAR
  I : INTEGER;
BEGIN RESET(Fic);
FOR I:=0 TO 127 DO READ(Fic,Tab[I]);
END;


{=============================================================================}
     PROCEDURE EcritureImage(VAR Fic : FichierPlan;VAR Tab : Plan);
{=============================================================================}

VAR
  I : INTEGER;

BEGIN REWRITE(Fic);
FOR I:=0 TO 127 DO WRITE(Fic,Tab[I]);
END;


{*****************************************************************************}
END.                          {FIN DU FICHIER}
{*****************************************************************************}
```